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

ТРУДЫ НИИСИ РАН

ТОМ 9 № 3

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ:

ТЕОРЕТИЧЕСКИЕ И ПРИКЛАДНЫЕ АСПЕКТЫ

МОСКВА
2019

Тематика номера:
Исследование физических процессов и их моделирование, информационные и компьютерные технологии, визуализация, робототехника, медицинские и экономические исследования.

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:

Research and modeling of physical processes, Information and computer technologies, Visualization, Robotics, Medical and Economic research.

The Journal publishes novel articles on the following research areas: 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: В.Е.Текунов

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

СОДЕРЖАНИЕ

Изменения термодинамических свойств двумерной модели Изинга при переходе от 4 к 6 связям на спин

М.Ю. Мальсагов, Б.В. Крыжановский, Я.М. Карандашев

ФГУ ФНЦ НИИСИ РАН, г. Москва
{malsagov, kryzhanov, karandashev}@niisi.ras.ru

Аннотация: В настоящей работе мы исследовали как изменится плотность состояний в 2D модели Изинга при включении взаимодействия со следующими соседями. Иными словами, мы рассматриваем случай двумерной решётки с диагональными связями. В такой модели каждый спин имеет по 6 связей, как и в 3D решётке. Благодаря планарности модели есть возможность точного вычисления статистической суммы и прочих характеристик этой модели при помощи полиномиального алгоритма. В работе проводится численное моделирование при помощи алгоритма Кастелейна-Фишера, которое позволило исследовать, как изменяются критические величины и плотность состояний при включении дальнодействия. Полученные результаты позволяют утверждать, что включение рассмотренного здесь типа дальнодействия изменяет только количественные характеристики системы, не влияя на качественные. Однозначного результата при конечных размерах решётки нельзя дать, но есть все основания полагать, что имеет место логарифмическая расходимость теплоёмкости в критической точке.

Ключевые слова: Нормировочная константа, плотность состояний, двумерная решётка с диагональными связями, взаимодействие с ближайшими соседями, критическая температура, пик теплоёмкости, спектральная функция.

1. Введение

Термодинамические свойства спиновых систем представляют интерес не только в физике, но и в ряде других областей науки, техники и информатике. Особый интерес представляет собой двумерная модель Изинга, которая часто рассматривается как модель изображения в задачах компьютерного зрения. Для нее получено аналитическое решение для решетки бесконечных размеров [1]. Кроме того, существуют такие алгоритмы, как например, алгоритм Кастелейна-Фишера [2-5], позволяющих получить точные численные значения свободной энергии и маргинальной вероятности для любой конечной размерности решётки и при любых коэффициентах взаимодействия.

Основополагающей характеристикой, определяющей свойства спиновой системы, является плотность состояний $D(E)$, представляющая собой вырождение уровней энергии. Знание распределения энергий позволяет легко вычислить статистическую сумму и все вытекающие из этого свойства системы. Действительно, рассмотрим систему из N спинов $s_i = \{\pm 1\}$, $i = \overline{1,N}$, на квадратной решетке $N = L \times L$. Энергия системы в конфигурационном состоянии $S = (s_1, s_2, ..., s_N)$ задается выражением:

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

При заданной температуре, вероятность того, что система находится в состоянии s, описывается формулой распределения Гиббса:

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

Статистическая сумма такой системы представима в виде

$$Z = \sum\nolimits_E D(E) e^{-N\beta E} , \quad (2)$$

где β - обратная температура, а суммирование ведется по всем значениям энергии. Вместо константы Z удобнее рассматривать (удельную) свободную энергию системы:

$$f(\beta) = -N^{-1} \ln Z , \quad (3)$$

Знание свободной энергии, позволяет вычислять основные измеряемые физические параметры системы:

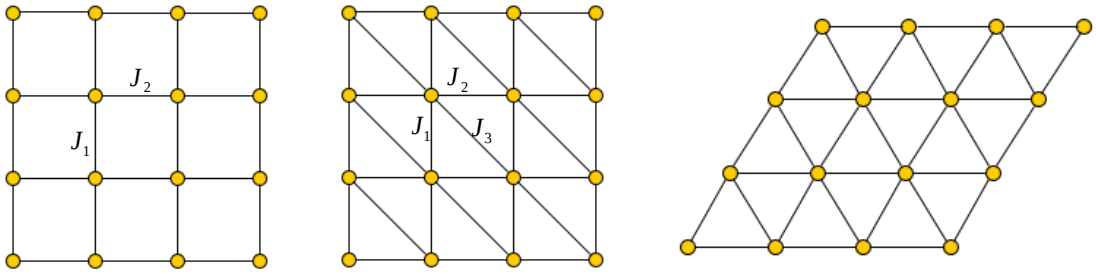
$$U = \frac{df}{d\beta}, \sigma^2 = -\frac{d^2 f}{d\beta^2}, C = -\beta^2 \frac{d^2 f}{d\beta^2}, \quad (4)$$

где внутренняя энергия $U = \langle E \rangle$ есть среднее по ансамблю при заданном β, $\sigma^2 = \langle E^2 \rangle - \langle E \rangle^2$ - дисперсия энергии, а

$C = \beta^2\sigma^2$ - удельная теплоемкость.

В настоящей работе мы рассматриваем случай двумерной решётки с диагональными связями, изображенной на рисунке 1. Помимо четырёх связей, идущих от каждого спина, как в стандартной 2D модели Изинга (рис.1а), мы добавляем ещё две связи, идущие по диагонали (рис. 1б). Выбор такой модели обусловлен следующим: с одной стороны, такая решётка при добавлении связей J_3 по-прежнему остаётся планарной и к ней применим алгоритм Кастелейна-Фишера; с другой стороны, каждый спин в ней имеет по 6 связей, как и в 3D решётке.

В последующих разделах мы исследуем изменение критических значений величин (3)-(4) при изменении величины диагональной связи от $J_3 = 0$ до $J_3 = 1$. Особое внимание будет уделено изменению спектра энергий, т.е. плотности состояний $D(E)$.



(a) (b) (c)

Рис. 1. Стандартная прямоугольная решётка (a) имеет по четыре связи, идущие от каждого спина (J_1 - вертикальные веса связей, J_2 - горизонтальные). При добавлении диагоналей J_3 (b) она превращается в треугольную решётку (c) с шестью связями на спин.

2. Исходные выражения

Известно, что в простейшем случае прямоугольной решётки, когда взаимодействие с ближайшими четырьмя соседями задается константой $J_{ij} = J = 1$, известно аналитическое решение Онсагера [1], полученное в асимптотическом пределе $N \to \infty$ с периодическими граничными условиями, которое имеет вид:

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

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

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

где

$$k = \frac{2\sinh 2\beta}{\cosh^2 2\beta}, \qquad (8)$$

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

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

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

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

Связь величин (5)-(7) с плотностью состояний подробно проанализирована в работе [6]. Введем здесь понятие спектральной функции:

$$\Psi(E) = \frac{1}{N}\ln D(E). \qquad (11)$$

Функция $\Psi(E)$ связана с величиной свободной энергии $f(\beta)$ соотношениями Лежандра:

$$\Psi(E) = \beta E - f(\beta), \quad E = \frac{df(\beta)}{d\beta}, \qquad (12)$$

которые мы использовали для пересчета данных эксперимента. Кроме того, мы использовали вытекающие из (12) выражения [6] для производных спектральной функции, необходимые при анализе спектральной плотности:

$$\Psi' = \beta$$
$$\Psi'' = \left(\frac{d^2 f}{d\beta^2} \right)^{-1} \qquad (13)$$
$$\Psi''' = -\frac{d^3 f}{d\beta^3} \cdot \left(\frac{d^2 f}{d\beta^2} \right)^{-3}$$

Здесь и далее мы используем обозначения $\Psi' = d\Psi / dE$, $\Psi'' = d^2\Psi / dE^2$ и $\Psi''' = d^3\Psi / dE^3$. Подчеркнем, что в выражениях (12)-(13) величину β следует понимать как свободную переменную [6], а не обратную температуру. Если же β есть обратная температура, то в (13) следует подставить $E = U$.

Как было показано в работе [7-8] для решёток конечной размерности логарифмической расходимости не наблюдается. Однако высота пика дисперсии близи β_c растёт логарифмически с ростом размерности решётки. В то же время для кубических решёток размерностей больших, чем 2D, получить аналитического решения критических показателей так и не удалось – численное моделирование показало [9], что фазовый переход в кубических решётках размерностью начиная с 5D (5D, 6D и т.д.) характеризуется конечным скачком, а не расходимостью теплоемкости как в 2D случае. Что же касается трёхмерной и четырехмерной решёток – то здесь вопрос до сих пор открыт.

В работе [10] методом n-окрестностей было показано, что конечный скачок теплоёмкости при фазовом переходе имеет место лишь когда количество связей спина превышает некоторую величину $q > 16/3$. Таким образом, для 3D решётки, в которой число связей удовлетворяет этому неравенству, потенциально возможен конечный скачок теплоёмкости. Однако этот результат скорее является некорректностью упомянутого метода n-окрестностей.

В заключение этого раздела приведем выражение для энергии основного состояния E_0, учитывающее конечные размеры решетки и свободные граничные условия:

$$E_0 = \left(1 - L^{-1}\right)\left[J_1 + J_2 + J_3(1 - L^{-1}) \right] \approx$$
$$\approx 2(1 + 0.5 J_3) \qquad (14)$$

J_3	$\beta_{\max}$	$f_{\max}$	$U_{\max}$	$\sigma^2_{\max}$	$\Psi_{\max}$	$\Psi'_{\max}$	$\Psi''_{\max}$
0.0	0.4600	-0.9348	-1.2937	5.3312	0.3397	0.4600	-0.1876
0.1	0.4200	-0.9068	-1.2678	5.9875	0.3743	0.4200	-0.1670
0.2	0.4050	-0.9088	-1.3683	6.8113	0.3547	0.4050	-0.1468
0.3	0.3800	-0.8956	-1.3846	7.5492	0.3695	0.3800	-0.1325
0.4	0.3600	-0.8874	-1.4167	8.2920	0.3774	0.3600	-0.1206
0.5	0.3500	-0.8922	-1.5195	9.1002	0.3604	0.3500	-0.1099
0.6	0.3400	-0.8963	-1.6162	9.7898	0.3468	0.3400	-0.1021
0.7	0.3200	-0.8830	-1.5994	10.6699	0.3712	0.3200	-0.0937
0.8	0.3100	-0.8845	-1.6732	11.4568	0.3659	0.3100	-0.0873
0.9	0.3000	-0.8851	-1.7380	12.2417	0.3636	0.3000	-0.0817
1.0	0.2900	-0.8845	-1.7933	13.0417	0.3645	0.2900	-0.0767

Таблица 1. Критические значения измеряемых величин при различных J_3.

3. Эксперимент

Для проведения экспериментов мы использовали алгоритм Кастелейна-Фишера [2-5], 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 дисперсия σ^2) для ряда решёток различной размерности $N = L \times L$, $L = 10 \div 200$. 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.

В ходе экспериментов величина связи J_3 варьировалась от 0 до 1. Для каждого значения J_3 установлены зависимости $f = f(\beta)$, $U = U(\beta)$ и $\sigma^2 = \sigma^2(\beta)$. С учетом (12)-(13) вычислена плотность состояний: построена функция $\Psi = \Psi(E)$ и её производные. По положению максимума кривой $\sigma^2 = \sigma^2(\beta)$ определялась критическая температура $\beta_{\max}$ и значения величин $f_{\max} = f(\beta_{\max})$, $U_{\max} = U(\beta_{\max})$ и $\sigma^2_{\max} = \sigma^2(\beta_{\max})$. Также определены величины $\Psi_{\max} = \Psi(U_{\max})$, $\Psi'_{\max} = \Psi'(U_{\max})$, $\Psi''_{\max} = \Psi''(U_{\max})$ и $\Psi'''_{\max} = \Psi'''(U_{\max})$. Данные экспериментов приведены в Таблице 1 и показаны на рисунках 2-12.

Ниже мы приводим данные только для решетки размера $N = 18\times18$, поскольку для решеток больших размеров кривые практически сливаются. Однако подчеркнем, что установленные ниже зависимости были одинаковы для всех значений $L = 10 \div 200$.

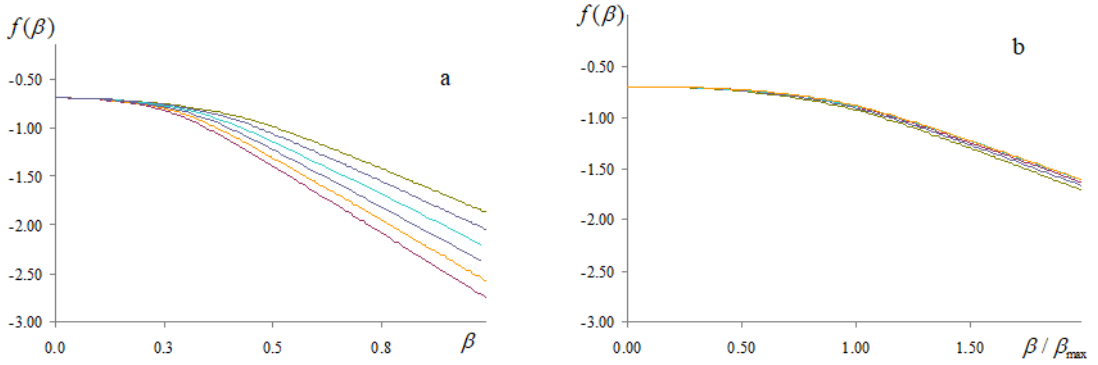


Рис. 2. Зависимость $f = f(\beta)$: (a) сверху вниз J_3=0, 0.2, 0.4, 0.6, 0.8 и 1; (b) те же кривые, но по оси абсцисс отложено $\beta / \beta_{\max}$.

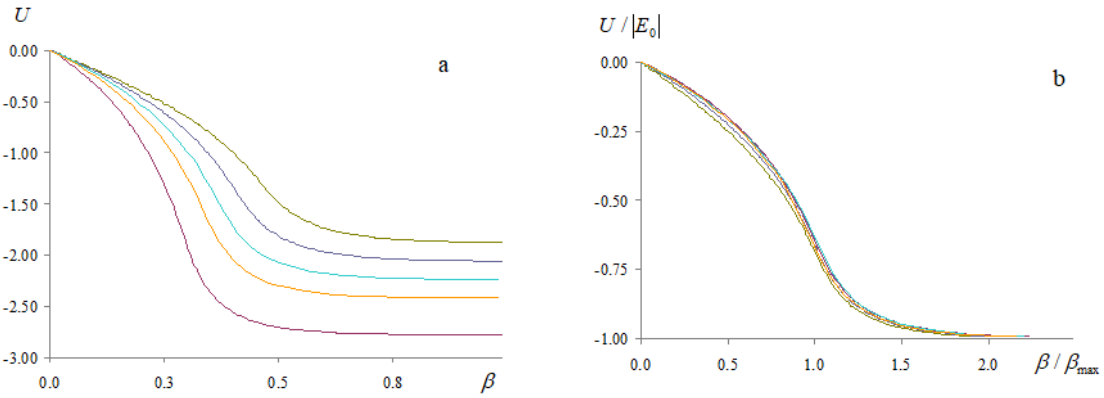


Рис. 3. Зависимость $U = U(\beta)$: (a) сверху вниз J_3=0, 0.2, 0.4, 0.6, 0.8 и 1; (b) те же кривые в масштабе $U(\beta)/|E_0|$ и $\beta / \beta_{\max}$.

3.1. Температурные зависимости

На рисунке 2a показано, как изменяется зависимость $f = f(\beta)$ при увеличении связи J_3 от 0 до 1. Показанные изменения были ожидаемы. Однако приведенные на рисунке 2b кривые показывают, что все эти изменения сводятся, в основном, к масштабированию оси абсцисс. То же самое можно сказать и о характере зависимости $U = U(\beta)$, показанной на рис.3. Масштабированные кривые на рис.3b (по оси ординат отложена величина $U/|E_0|$, а по оси абсцисс $\beta / \beta_{\max}$) хотя и не сливаются, однако отличаются весьма незначительно.

Изменения характера зависимости $\sigma^2 = \sigma^2(\beta)$ при изменении связи J_3 показаны

на рисунке 4. На рис. 4a видно, что с ростом J_3 пик кривой смещается в сторону меньших β, т.е. уменьшается величина $\beta_{\max}$. Одновременно, высота пика возрастает, т.е. возрастает величина $\sigma^2_{\max}$. Масштабирование кривых, показанное на рис. 4b, приводит к тому, что при $\beta > 0.5\beta_{\max}$ кривые для всех значений J_3 практически сливаются. Тем самым создается впечатление, что термодинамические свойства системы качественно слабо зависят от величины диагональной связи J_3 и учет следующих соседей сводится к простому масштабированию. Однако ниже мы увидим, что спектральные характеристики изменяются весьма заметно.

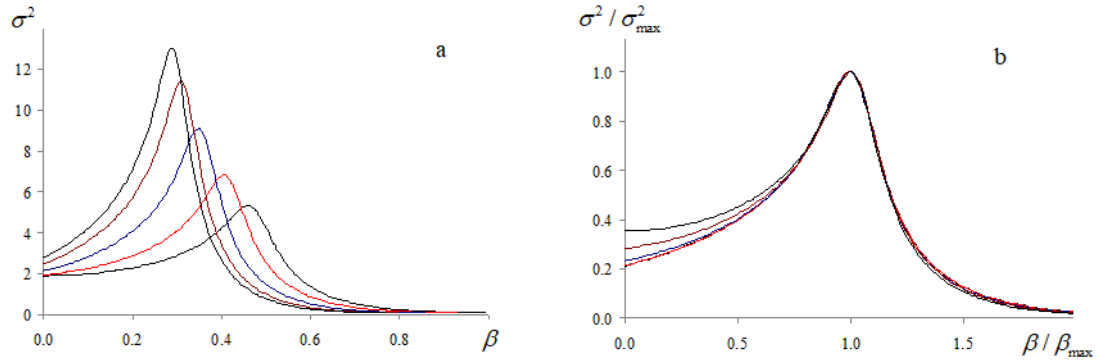


Рис. 4. Зависимость $\sigma^2 = \sigma^2(\beta)$: (a) справа налево J_3=0, 0.2, 0.4, 0.8 и 1; (b) те же кривые, но в масштабе $\sigma^2 / \sigma^2_{\max}$ и $\beta / \beta_{\max}$.

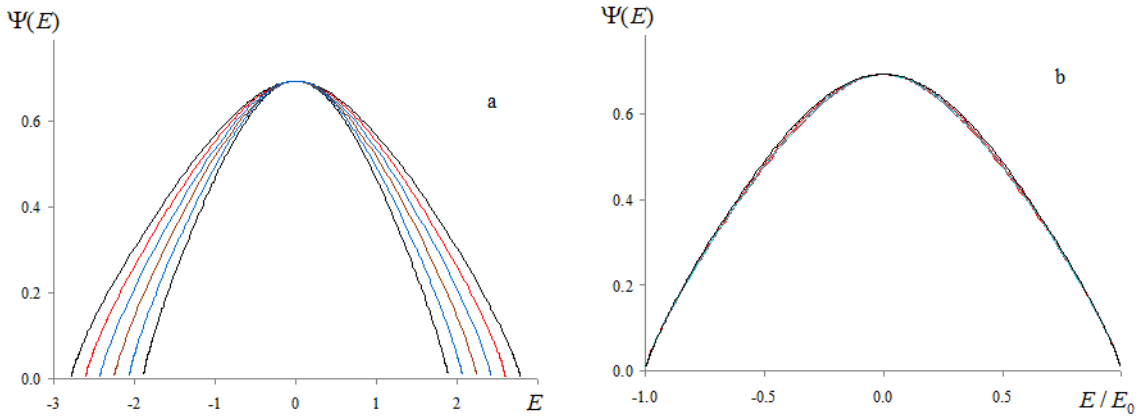


Рис. 5. Зависимость $\Psi = \Psi(E)$: (a) снизу вверх J_3=0, 0.2, 0.4, 0.6, 0.8 и 1; (b) те же кривые, но по оси абсцисс отложена нормированная величина E / E_0.

3.2. Спектральные характеристики

В ходе эксперимента измерялась свободная энергия и на основании выражений (11)-(13) вычислялась плотность состояний. На рис. 5 показаны графики спектральной функции $\Psi(E)$ для разных значений диагональной связи J_3. При масштабировании оси абсцисс (рис. 5b) кривые практически сливаются. Однако, графики для первой (рис. 6) и второй (рис. 7) производных спектральной функции $\left(\Psi' = d\Psi / dE, \Psi'' = d^2\Psi / dE^2\right)$ различаются очень заметно.

На рис. 7 видно, что с ростом J_3 кривая $\Psi'' = \Psi''(E)$ приподнимается (возрастает критическое значение $\Psi''_{\max}$), а плавный ход кривой в центре ($E = 0$) сменяется своеобразным «клювом». Это сильно сказывается на поведении третьей производной $\Psi''' = d^3\Psi / dE^3$, показанном на рис. 8. В точке $E = 0$ кривая $\Psi'''(E)$ проходит через ноль только при $J_3 = 0$. Небольшое увеличение величины J_3 приводит к тому, что

переход через точку $E=0$ сопровождается скачком $\Psi'''(E)$. Максимальный скачок имеет место при $J_3=1$. Интересно, что поведение кривых в зависимости от J_3 разбивается на два класса. Первый класс – это кривые с $J_3 \le 0.3$. Кривые этого класса характеризуются экстремумом величины $|\Psi'''(E)|$ в окрестности точки $E=0$. Кривые второго класса ($J_3 > 0.3$) такого экстремума не имеют. Интересно, что на границах спектра (при $|E| \to |E_0|$) кривые каждого из классов практически сливаются (см. рис. 8b), образуя два асимптотических класса.

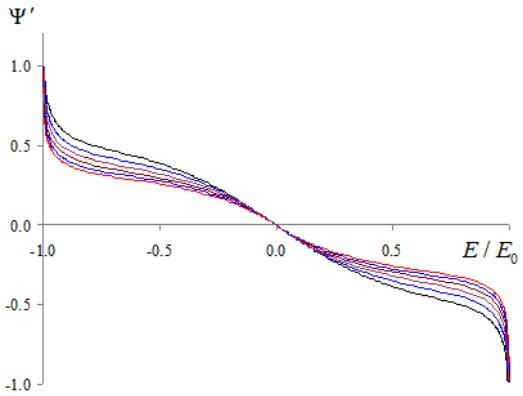


Рис.6. Зависимость $\Psi' = \Psi'(E)$ при J_3=0, 0.2, 0.4, 0.6, 0.8, 1 (сверху вниз).

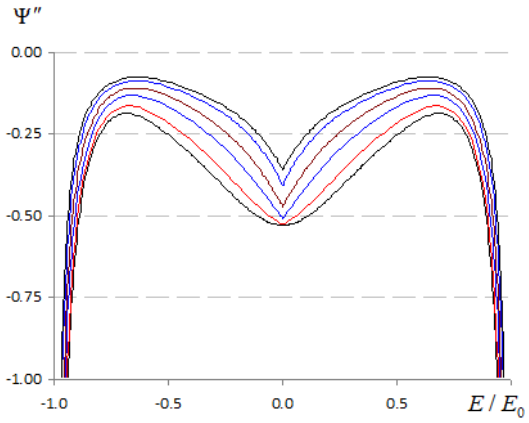
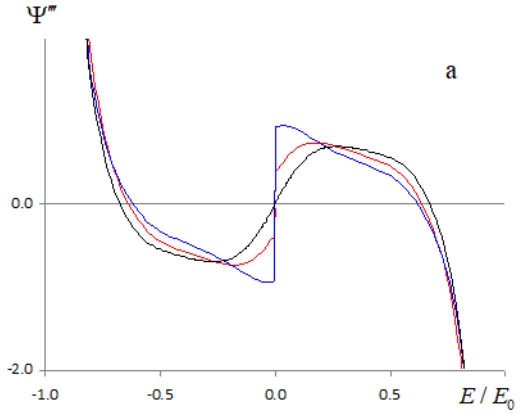


Рис.7. Зависимость $\Psi'' = \Psi''(E)$ при J_3= 0, 0.2, 0.4, 0.6, 0.8, 1 (снизу вверх).



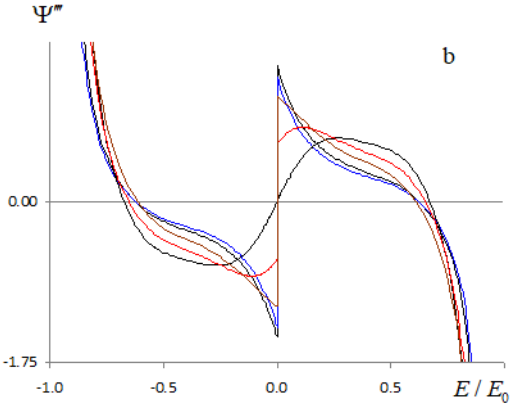


Рис.8. Зависимость третьей производной $\Psi''' = \Psi'''(E)$: (a) - кривые для J_3=0, 0.2, 0.3; (b) – кривые для J_3=0, 0.2, 0.4, 0.6, 0.8 и 1. На обоих рисунках без скачка через нуль проходит только кривая с J_3=0. Наибольший скачок в точке $E=0$ имеет место при $J_3=1$. На границах спектра кривые с $J_3 \le 0.3$ и кривые с $J_3 > 0.3$ образуют два асимптотических класса.

3.3. Критические величины

Анализ экспериментальных данных позволил вывести ряд аналитических выражений, характеризующих поведение системы и зависимость критических величин от величины связи J_3 со следующими соседями.

На рисунке 9 показана зависимость критической температуры $\beta_{\max}$ от величины J_3. Было замечено, что величина $\beta_{max}|E_0|$ линейным образом зависит от J_3 в виде $\beta_{max}|E_0| \approx 0.8475 - 0.0368\,J_3$. С учетом (14) для $\beta_{\max}$ получим:

$$\beta_{max} \approx 1.044 \cdot \beta_{ONS} \frac{1-0.047\,J_3}{1+0.5J_3}, \qquad (15)$$

где 1.044 - поправочный множитель, учитывающий конечные размеры системы [7]. Убирая этот поправочный коэффициент, получаем выражение для асимптотически больших решеток:

$$\beta_c = \lim_{L\to\infty} \beta_{\max} \approx \beta_{ONS} \frac{1-0.047\,J_3}{1+0.5J_3}, \qquad (16)$$

Нетрудно заметить, что при $J_3=0$ это выражение совпадает с результатом (10) для

классической 2D модели Изинга, а при $J_3 = 1$ выражение (16) с точностью до 0.5% совпадает с известным выражением из [11] для треугольной решетки $\beta_c \approx 0.2747$.

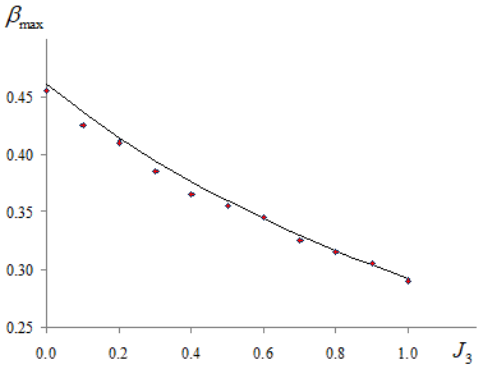


Рис.9. Зависимость $\beta_{max} = \beta_{max}(J_3)$.

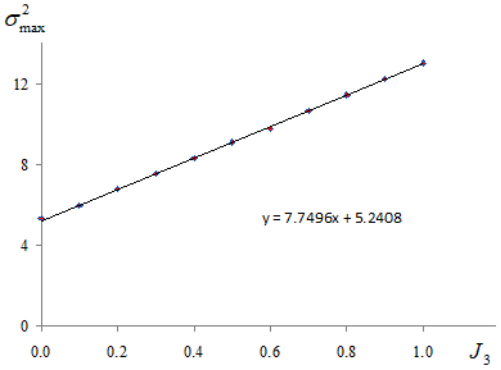


Рис.10. Зависимость $\sigma^2_{max} = \sigma^2_{max}(J_3)$.

Графики зависимости величин σ^2_{max}, Ψ'_{max} и Ψ''_{max} от величины J_3 представлены на рисунках 10, 11 и 12. Представленные зависимости можно с хорошей точностью аппроксимировать выражениями:

$$\sigma^2_{max} \approx 5.24 + 7.75 \cdot J_3, \qquad (17)$$

$$\dot{\Psi}_{max} \approx 0.44 - 0.16 \cdot J_3, \qquad (18)$$

$$\ddot{\Psi}_{max} \approx \frac{0.128}{0.676 + J_3}. \qquad (19)$$

Как видим, величины σ^2_{max} и Ψ'_{max} линейно связаны с J_3.

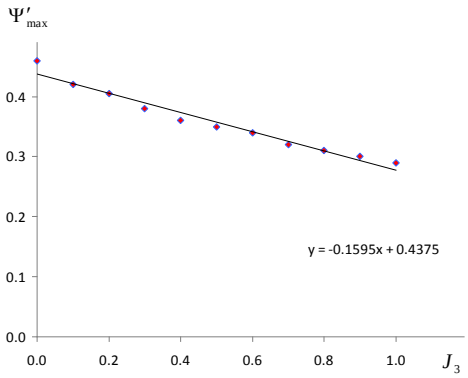


Рис.11. Зависимость $\Psi'_{max} = \Psi'_{max}(J_3)$

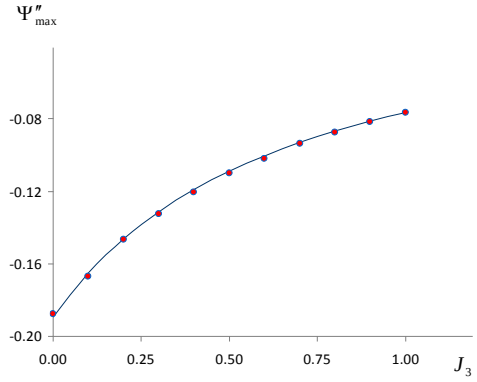


Рис.12. Зависимость $\Psi''_{max} = \Psi''_{max}(J_3)$

3.4. Критические величины от размерности решетки

В дальнейших экспериментах мы остановились на исследовании свойств треугольной решётки (при $J_1 = J_2 = J_3 = 1$) при увеличении размерности решётки L (рис. 13-14, табл. 2). На рис. 13 видно, что с ростом размера решётки меняется положение пика дисперсии – пик становится уже и сдвигается влево, в сторону более высоких температур. На рис. 14 и в табл. 1 показаны пиковые значения дисперсии и критической температуры для разных размеров решётки.

Известно, что для прямоугольной решётки с четырьмя связями уравнение для критической температуры имеет вид:

$$sh(2\beta_c) = 1, \qquad (20)$$

откуда следует (10). Для треугольной решётки с шестью связями уравнение принимает вид:

$$sh(2\beta_c) = \frac{1}{\sqrt{3}}, \qquad (21)$$

решение которого даёт значение $\beta_c \approx 0.2747$ [11], очень близкое к найденному в эксперименте (см. табл. 2).

Как следует из полученных данных, высота пика дисперсии энергии растёт логарифмически с ростом размера решётки:

$$\sigma^2_{\max} \approx -5.3394 + 6.3808 \ln L, \qquad (22)$$

в то время как положение пика плавно подходит к своему асимптотическому значению:

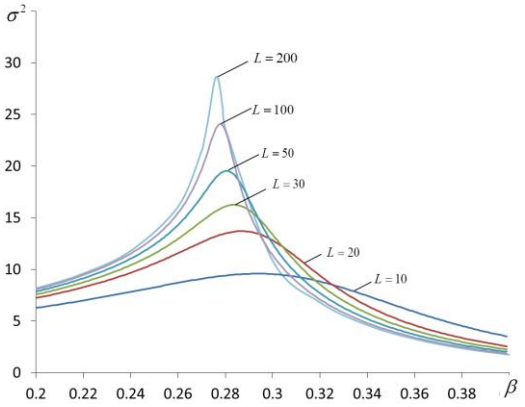


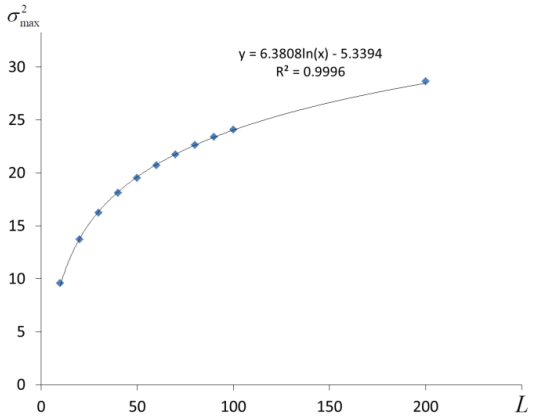
Рис. 13. Зависимость дисперсии энергии σ^2 от обратной температуры β для разных значений размера треугольной решётки $L = 10...200$.

$$\beta_{\max} \approx 0.275 + \frac{0.3089}{L} - \frac{1.1911}{L^2}. \qquad (23)$$

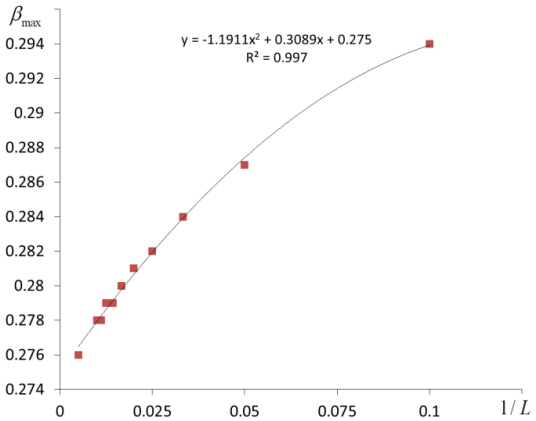
Основной вывод в том, что, вероятно, такой же результат ожидаем и для 3D модели, обладающей тем же количеством связей на один спин, что и треугольная решётка.

L	$\sigma^2_{\max}$	$\beta_{\max}$
10	9.58732	0.294
20	13.7079	0.287
30	16.2503	0.284
40	18.0956	0.282
50	19.5381	0.281
60	20.7306	0.28
70	21.7444	0.279
80	22.6089	0.279
90	23.3964	0.278
100	24.0952	0.278
200	28.6198	0.276

Таблица 2. Пиковые значения дисперсии энергии и обратной температуры для различных размеров треугольной планарной решётки.



(a)



(b)

Рис. 14. Зависимость (a) максимального значения дисперсии энергии $\sigma^2_{\max}$ и (b) положения пика $\beta_{\max}$ от размера решётки $L = 10...200$. (На (b) по оси абсцисс отложен обратный размер решётки $1/L$). Маркеры обозначают экспериментальные значения, сплошные кривые дают линию тренда, уравнение которого также дано на рядом с графиком.

4. Заключение

Мы исследовали, как изменится плотность состояний в 2D модели Изинга при включении взаимодействия со следующими соседями. Анализ полученных данных показал, что термодинамические свойства системы качественно слабо зависят от величины диагональной связи J_3 и учет следующих соседей сводится к простому масштабированию. Сказанное во многом справедливо и для плотности состояний. Действительно, глядя на рис. 5 нетрудно заметить, что в масштабированных кривых спектральной плотности $\Psi(E)$ различия при разных J_3 практически не заметны. В то же время графики функций $\Psi'(E)$ и $\Psi''(E)$ существенным образом изменяются при изменении J_3. А именно поведением функции $\Psi''(E)$ определяются термодинамические свойства спиновой системы. Отсюда вытекает основной вывод из полученных результатов: вычисление плотности состояний с помощью известных алгоритмов (например, алгоритм

Ванга-Ландау [12]) и последующее вычисление статистической суммы не гарантируют корректное описание критического поведения системы даже при большой точности указанных алгоритмов.

Changes in the thermodynamic properties of the two-dimensional Ising model during the transition from 4 to 6 bonds per spin

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

Abstract. We analyzed a change in the density of states of a two-dimensional Ising model during the transition from a rectangular grid to a triangular grid when a next-next-neighbor interaction is introduced. In other words, we examined two-dimensional lattices with diagonal connections. The same as in a three-dimensional model in this case each spin has 6 connections. Since the model is planar, we can calculate the free energy and other characteristics of the system using a polynomial algorithm. We performed computer simulations using the Kasteleyn-Fisher algorithm, which allowed us to study changes of critical values and the density of states when a long-range interaction is taken into account. From the obtained results it follows that the interactions of the type analyzed here result in quantitative changes of the system's characteristics, but they did not change them qualitatively. It is shown that with increasing N the heat capacity at the critical point increases logarithmically.

Keywords: normalization constant, partition function, Ising model, density of states, two-dimensional lattice with diagonal connections, critical temperature, dispersion peak, spectral function

Литература

1. L. Onsager. A two-dimensional model with an order–disorder transition. "Crystal statistics. I. Phys. Rev". 65 (3–4), 117–149, 1944.
2. P. Kasteleyn. Dimer statistics and phase transitions. "J. Math. Phys.", 4(2), 1963.
3. M. Fisher. On the dimer solution of planar Ising models. "J. Math. Phys.", 7(11), 1966.
4. Ya.M. Karandashev, M.Yu. Malsagov. Polynomial algorithm for exact calculation of partition function for binary spin model on planar graphs. "Optical Memory and Neural Networks (Information optics)", v. 26, №2, 2017. https://arxiv.org/abs/1611.00922
5. N. Schraudolph and D. Kamenetsky. Efficient exact inference in planar Ising models. "NIPS-2008", 2008. https://arxiv.org/abs/0810.4401
6. B.V. Kryzhanovsky. Features of the Spectral Density of a Spin System. "Doklady Mathematics", vol. 97, №2, (2018), 188-192.
7. 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)", v. 26, №3, (2017), 165–171.
8. B.V. Kryzhanovsky, M.Yu. Malsagov, and I.M. Karandashev. Dependence of Critical Parameters of 2D Ising Model on Lattice Size. "Optical Memory and Neural Networks (Information Optics)", v. 27, №1, (2018), 10–22.
9. P.H. Lundow, K. Markström. The discontinuity of the specific heat for the 5D Ising model. "Nuclear Physics B", v. 895, (2015), 305-318. https://doi.org/10.1016/j.nuclphysb.2015.04.013
10. L. Litinskii and B. Kryzhanovsky. Spectral density and calculation of free energy. "Physica A: Statistical Mechanics and its Applications", v. 510, C, (2018), 702-712.
11. R.J.Baxter. Exactly solved models in statistical mechanics. London, Academic Press, 1982.
12. Fugao Wang and D. P. Landau. Efficient, Multiple-Range Random Walk Algorithm to Calculate the Density of States. "Phys. Rev. Lett.", v. 86, 2050, 2001.

Спектральная плотность спин-стекольной модели Эдвардса-Андерсона

Я.М. Карандашев, М.Ю. Мальсагов, Б.В. Крыжановский

ФГУ ФНЦ НИИСИ РАН, г. Москва
{karandashev, malsagov, kryzhanov}@niisi.ras.ru

Аннотация: В настоящей работе мы исследовали плотность состояний модели Эдвардса-Андерсона и получили аппроксимационную формулу, которая даёт хорошее согласие с результатами, полученными в численном эксперименте. Важно, что полученная аппроксимация даёт отличное совпадение не только самой функции плотности состояний, но и её первой и второй производных, которые наиболее ценны для получения критических параметров системы. Полученные оценки могут позволить в дальнейшем исследовать двумерные изинговые модели с точки зрения поведения системы при различных температурах, в том числе для задач байесового вывода и в задачах обучения.

Ключевые слова: модели Эдвардса-Андерсона, спиновое стекло, глобальное состояние, спектр энергий.

1. Введение

Вычисление статистической суммы является центральной задачей статистической физики и информатики. Для целого ряда содержательных моделей удается решить эту задачу точно [1]. Из всех моделей следует выделить 2D модель Изинга [2], которая, несмотря на свою простоту, имела решающее значение для изучения критических явлений. Следует упомянуть также модели Эдвардса-Андерсона [3] и Шеррингтона-Киркпатрика [4], внесшие большой вклад в развитие теории спиновых стекол. Однако точно решаемых моделей совсем немного. Поэтому для анализа поведения сложных систем используются численные методы, из которых мы выделим два наиболее продвинутых. Первый из них – это метод Монте-Карло [5-6], позволяющий наиболее полно исследовать измеряемые величины и устанавливать критические параметры системы [7-8]. Детальный анализ точности метода и его возможностей проведен в работах [12-13]. К сожалению, этот метод требует большого объема вычислений и не позволяет впрямую вычислять свободную энергию. Второй метод основан на идеях, заложенных в работах [14-15], которые недавно реализованы в виде быстрого алгоритма [16-17]. Суть алгоритма в том, что вычисление свободной энергии сводится к вычислению детерминанта некоторой матрицы. Этот алгоритм активно используется, поскольку вычисление свободной энергии производится с машинной точностью и одновременно можно определять энергию основного состояния и его конфигурацию.

В данной работе исследуется спектр спиновой системы конечных размеров с матрицей взаимодействий модели Эдвардса-Андерсона, гамильтониан которой задается в виде квадратичного функционала (1). Данный функционал довольно часто встречается в задачах машинного обучения и задачах обработки изображений. Величины имеют смысл принадлежности к одному из двух классов (фон/объект) пикселей на картинке, либо являются активностью нейронов в байесовой нейронной сети, поэтому исследование подобных систем может открыть дорогу к новым алгоритмам машинного обучения.

2. Спектральная плотность

Рассмотрим систему, описываемую гамильтонианом:

$$E(S) = -\frac{1}{2N}\sum_{i,j=1}^{N} J_{ij} s_i s_j \ .$$

Это система из N изинговых спинов $s_i = \pm 1$ ($i = 1, 2, ..., N$), расположенных в узлах планарной решетки, узлы которой пронумерованы индексом i.

Мы рассматриваем планарную модель Эдвардса-Андерсона (EA-модель), в которой спины расположены в узлах двумерной решётки, каждый спин имеет ненулевые связи лишь с ближайшими четырьмя соседями, а парные взаимодействия J_{ij} имеют нормальное распределение $\left(\bar{J} = 0,\ \sigma_J = 1\right)$.

Нас интересует свободная энергия системы:

$$f = -\frac{1}{N}\ln Z,\ Z = \sum_S e^{-N\beta E(S)},$$

где статистическая сумма Z определена как сумма по всем возможным конфигурациям S, а $\beta = 1/kT$ - обратная температура. Знание свободной энергии позволяет вычислять основные измеряемые параметры системы:

$$U = \frac{\partial f}{\partial \beta},\ \sigma^2 = -\frac{\partial^2 f}{\partial \beta^2},\ C = -\beta^2 \frac{\partial^2 f}{\partial \beta^2}, \quad (1)$$

где внутренняя энергия $U = \langle E \rangle$ есть среднее по ансамблю при заданном β, $\sigma^2 = \langle E^2 \rangle - \langle E \rangle^2$ - дисперсия энергии, а $C = \beta^2\sigma^2$ - удельная теплоемкость. В работах [18-20] проведен подробный анализ величин (3) в зависимости от размера решётки и при включении небольшого шума в матричные элементы при постоянном ненулевом среднем значении. В этой работе мы сконцентрируемся на анализе спектральной плотности модели Эдвардса-Андерсона, когда матричные элементы имеют нормальное распределение с нулевым средним. Пусть $D(E)$ есть число состояний с энергией E. Тогда статсумму можно представить в виде $Z = \sum_E D(E)e^{-N\beta E}$. Переходя здесь от суммирования к интегрированию с точностью до несущественной константы получим:

$$Z \sim \int_{-\infty}^{\infty} e^{N[\Psi(E)-\beta E]}dE, \quad (2)$$

где $\Psi(E) = \ln D(E)/N$. Оценив интеграл (4) методом седловой точки, получим $Z \sim \exp[-Nf(\beta)]$, где

$$f(\beta) = \beta E - \Psi(E),\ \frac{d\Psi(E)}{dE} = \beta. \quad (3)$$

Первое из выражений (5) определяет свободную энергию, а второе определяет значение E в седловой точке, где производная функции $\Psi(E) - \beta E$ обращается в ноль.

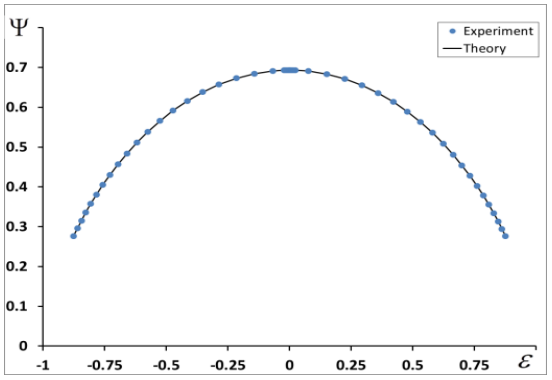
В работах [16-17] предлагаются алгоритмы для точного вычисления свободной энергии планарных решеток. Данный алгоритм позволил нам вычислять функцию $f = f(\beta)$ и её производные. Это в свою очередь позволяет исследовать распределение энергий $D(E) = \exp[N\Psi(E)]$. Действительно, из уравнений (5) несложно получить уравнения для спектральной функции:

$$\Psi(E) = \beta E - f(\beta),\ E = \frac{df}{d\beta} \quad (4)$$

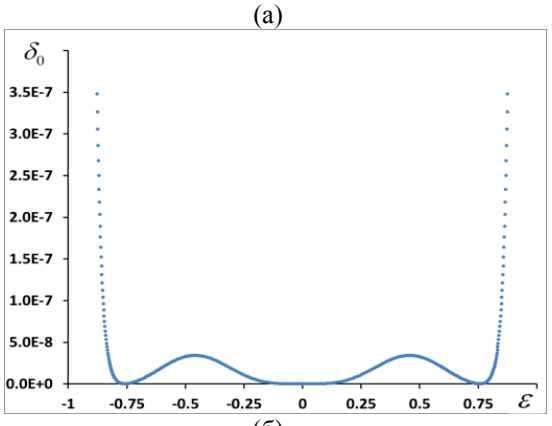
и ее производных

$$\frac{d\Psi}{dE} = \beta,\qquad \frac{d^2\Psi}{dE^2} = \left(\frac{d^2 f}{d\beta^2}\right)^{-1}. \quad (5)$$

Отметим, что $\Psi(E)$ с точностью до константы представляет собой энтропию, а уравнения (6) есть известные преобразования Лагранжа, применимые и для анализа спектральной плотности конечно размерных моделей [21-22]. Из этих уравнений вытекает следующее: когда величина β пробегает значения от $\beta = 0$ до $\beta = \infty$ величина E изменяется от 0 до E_0, и для каждого значения β мы имеем пару значений E и $\Psi(E)$. Тем самым мы прописываем вид функции $\Psi(E)$ и ее производных. Графики функции $\Psi(E)$ и её производных, построенные на основании экспериментальных данных, приведены на рис. 1-3. На графиках по оси абсцисс отложена величина $\varepsilon = E/|E_0|$.



(а)



(б)

Рис. 1. (а) - спектральная плотность. Сплошная линия – аналитическое выражение, маркеры – экспериментальные данные, отображены только каждая 10-ая точка. (б) – квадрат разности теории и экспериментальных данных в каждой точке.

3. Аппроксимация экспериментальных данных

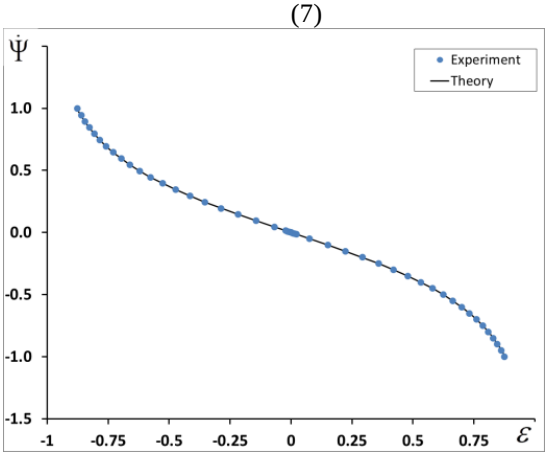
Получить спектральную плотность заданной системы практически невозможно, т.к. число состояний системы растет экспоненциально от размерности. Описать аналитическими выражениями достаточно точно также не представляется возможным. В предыдущем разделе мы получили спектральную плотность $\Psi(E)$, используя свободную энергию $f = f(\beta)$, которую можно точно посчитать экспериментально.

Анализируя полученное распределение $\Psi(E)$, мы пришли к аппроксимационной формуле:

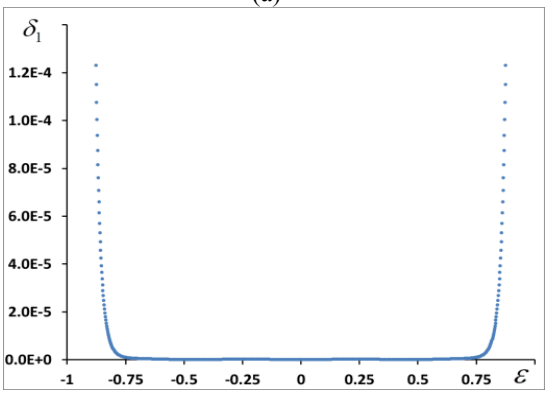
$$\Psi(\varepsilon) = \ln 2 - a\left[(1-b\varepsilon)\ln(1-c\varepsilon)\right] - a\left[(1+b\varepsilon)\ln(1+c\varepsilon)\right], \quad (6)$$

где

$$\varepsilon = E/|E_0|. \quad (7)$$



(а)



(б)

Рис. 2. (а) - первая производная спектральной плотности. Сплошная линия – аналитическое выражение, маркеры – экспериментальные данные, отображены только каждая 10-ая точка. (б) – квадрат разности теории и экспериментальных данных в каждой точке.

Варьируя свободные параметры a, b и c, можно определить их оптимальные значения:

$$a = 0.32,\ b = 1.19,\ c = 0.94, \quad (8)$$

при которых достигается наилучшее согласие (8) с данными эксперимента.

На рисунке 1а показано согласование экспериментальных данных (маркеры) с формулой (8) (сплошная кривая). Как видно, выражение (8) прекрасно описывает $\Psi(E)$. Средняя квадратичная ошибка аппроксимации (рис. 1б) на всем диапазоне составляет $\langle\delta\rangle \approx 3.4\cdot10^{-8}$. Наихудшее совпадение наблюдается на концах распределения, так как вблизи глобального минимума мало состояний и их трудно найти экспериментально. Чтобы визуализировать ошибку аппроксимации на рис.1б показан график величины ошибки δ_0, которая есть квадрат разности теории (8) и экспериментальных данных в каждой точке:

$$\delta_0 = \left[\Psi(E) - \Psi_{\exp}(E)\right]^2. \quad (9)$$

Хорошее согласие с экспериментальными данными получается также, если взять производную от выражения (8):

$$\frac{\partial\Psi}{\partial E} = -\frac{a}{|E_0|}\left[b\ln\frac{1+c\varepsilon}{1-c\varepsilon} + \frac{2c(b-c)\varepsilon}{1-c^2\varepsilon^2}\right]. \quad (10)$$

На рисунке 2а видно, что выражение (12) отлично описывает характер первой производной спектральной плотности без какой-либо доводки параметров или добавления новых. Ошибка аппроксимации (рис. 2б) на интервале $[-0.85; 0.85]$ не превышает $2\cdot10^{-5}$, а в центре распределения на 1-3 порядка меньше.

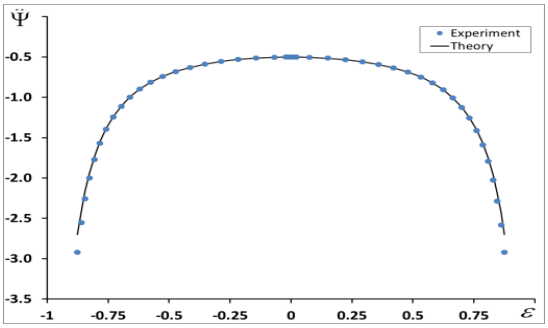
На рис.2б показан график величины ошибки δ_1, которая есть квадрат разности теории (12) и экспериментальных данных в каждой точке:

$$\delta_1 = \left[\frac{d\Psi}{dE} - \frac{d\Psi_{\exp}}{dE}\right]^2. \quad (11)$$

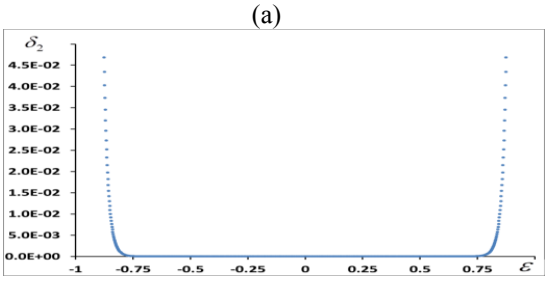
Аналогичный результат получается для второй производной:

$$\frac{\partial^2\Psi}{\partial E^2} = -\frac{2ac}{E_0^2}\cdot\frac{2b - c - c^3\varepsilon^2}{\left(1-c^2\varepsilon^2\right)^2}. \quad (12)$$

На рисунке 3а уже заметно влияние недостатка экспериментальных данных вблизи глобального минимума. Выражение (13) дается хорошее согласие в центральной части распределения, но на концах быстро ухудшается.



(а)



(б)

Рис. 3. (а) - вторая производная спектральной плотности. Сплошная линия – аналитическое выражение, маркеры – экспериментальные данные, отображены только каждая 10-ая точка. (б) – квадрат разности теории и экспериментальных данных в каждой точке.

На рис.3б показан график величины ошибки δ_2, которая есть квадрат разности теории (14) и экспериментальных данных в каждой точке:

$$\delta_2 = \left[\frac{d^2\Psi}{dE^2} - \frac{d^2\Psi_{\exp}}{dE^2} \right]^2 . \qquad (13)$$

Тем не менее, выражения (8) и (9) прекрасно описывают не только саму спектральную плотность, полученную с помощью (6), но и ее первые две производные (7). Стоит так же отметить, проведенные эксперименты для других размерностей показали, что при данной нормировке гамильтониана (1) вид спектральной плотности не зависит от размерности, т.е. графики сливаются. Поэтому выражение (8) можно применять для описания модели Эдвардса-Андерсона на планарной решетке любой размерности.

4. Заключение

Основываясь на экспериментальных данных точного вычисления свободной энергии, нам удалось построить спектральную плотность состояний модели Эдвардса-Андерсона на планарной решетке. Мы получили простое выражение (8), хорошо описывающее распределение спектральной плотности. Что более важно, аппроксимация оказалась настолько удачной, что позволила без дополнительно настройки свободных параметров получить приближения первой и второй производных спектральной плотности. Последнее обстоятельство очень важно: в работе [24] показано, что даже малейшие отклонения в спектральной плотности могут привести к кардинальным изменениям термодинамических свойств системы. Например, относительная разность спектральных плотностей в двумерных моделях Изинга и решетки Бете не превышает 1%. Однако, эти модели демонстрируют совершенно разные свойства: модель Изинга характеризуется расходимостью теплоемкости в критической точке, а у модели Бете в критической точке имеется конечный скачок теплоемкости.

В дальнейшем, используя наработки [23], возможно, удастся связать свободные параметры аппроксимации со статистическими характеристиками распределения элементов решетки. Мы надеемся, что полученные простые аппроксимационные формулы позволят создавать новые алгоритмы для задач компьютерной обработки изображений и других планарных задач на основе нейросетевого байесового вывода.

Density of states of Edwards-Anderson spin-glass model

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

Abstract. We obtain a simple approximation formula for the density of states of the Edwards-Anderson spin-glass model. This formula based on the results of numerical experiments and well approximate not only the density of states, but also its first and second derivatives, which are most valuable for obtaining the critical parameters of the system. The evaluations can be further used for examining the behavior of 2D Ising models at different temperatures, particularly for tackling Bayesian inference problems and learning algorithms.

Keywords: Edwards-Anderson model, spin glass, global state, density of states.

Литература

13. R.J.Baxter. Exactly solved models in statistical mechanics. London, Academic Press, 1982.
14. L.Onsager. Crystal statistics. A two-dimensional model with an order–disorder transition. "Phys. Rev.", 65 (3–4), (1944), 117–149.
15. S.F.Edwards, P.W.Anderson. Theory of spin glasses. "J. Phys. F: Metal Phys.", v. 5, (1975), 965.
16. D.Sherrington, P.Kirkpatrick. Solvable model of a spin-glass. "Phys. Rev. Lett.", v. 35 (26), (1975), 1792.
17. N.Metropolis, S. Ulam. The Monte Carlo Method. "J. of Am. Stat. Association", v. 44, № 247, (1949), 335—341.
18. George S. Fishman. Monte Carlo: concepts, algorithms, and applications. Springer, 1996.
19. Alex F. Bielajew. Fundamentals of the Monte Carlo method for neutral and charged particle transport. 2001.
20. W. M. C. Foulkes, L. Mitas, R. J. Needs and G. Rajagopal. Quantum Monte Carlo simulations of solids. "Reviews of Modern Physics", v. 73, (2001), 33.
21. J.W. Lyklema. Monte Carlo study of the one-dimensional quantum Heisenberg ferromagnet near = 0. "Phys. Rev. B", v. 27 (5), (1983), 3108–3110.
22. M.Marcu, J.Muller, F.-K.Schmatzer. Quantum Monte Carlo simulation of the one-dimensional spin-S xxz model. II. High precision calculations for S = ½. "J. Phys. A.", v. 18 (16), (1985), 3189–3203.
23. 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, №5, (2007), 653-755.
24. K.Binder. Finite Size Scaling Analysis of Ising Model Block Distribution Functions. "Z. Phys. B, Condensed Matter", v. 43, (1981), 119-140.
25. K.Binder, E.Luijten. Monte Carlo tests of renormalization-group predictions for critical phenomena in Ising models. "Physics Reports", v. 344, (2001), 179-253.
26. Kasteleyn P. Dimer statistics and phase transitions. "J. Math. Phys." v. 4(2). 1963.
27. Fisher M. On the dimer solution of planar Ising models. "J. Math. Phys.", v. 7(10). 1966.
28. Karandashev Ya.M., Malsagov M.Yu. Polynomial algorithm for exact calculation of partition function for binary spin model on planar graphs. "Optical Memory and Neural Networks (Information optics)", v. 26. №2, (2017), 87–95.
29. N.Schraudolph and D.Kamenetsky. Efficient exact inference in planar Ising models. "NIPS-2008", 2008. https://arxiv.org/abs/0810.4401.
30. B.V. Kryzhanovsky. M.Yu. Malsagov, I.M. Karandashev. Investigation of finite-size 2D Ising model with a noisy matrix of spin-spin interactions. "Entropy", v. 20 (8), (2018), 585.
31. B.V. Kryzhanovsky, I.M. Karandashev, M.Yu. Malsagov. Dependence of Critical Temperature on Dispersion of Connections in 2D Grid. "Advances in Neural Networks - ISNN 2018. Lecture Notes in Computer Science", v. 10878, (2018), 695-702.
32. I.M. Karandashev, B.V. Kryzhanovsky, M.Yu. Malsagov. Spectral characteristics of a finite 2D Ising model. "Optical Memory and Neural Networks (Information optics)", v. 27, №3, (2018), 147-151.
33. Häggkvist, R.; Rosengren, A.; Andrén, D.; Kundrotas, P.; Lundow, P.H.; Markström, K. Computation of the Ising partition function for 2-dimensional square grids. "Phys. Rev. E", 69, 046104, 2004.
34. Beale P.D. Exact distribution of energies in the two-dimensional Ising model. "Phys. Rev. Lett.", v. 76, (1996), 78–81.
35. Leonid Litinskii and Boris Kryzhanovsky. Spectral density and calculation of free energy. "Physica A: Statistical Mechanics and its Applications", v. 510, (2018), 702-712.
36. B.V. Kryzhanovsky. Features of the Spectral Density of a Spin System. "Doklady Mathematics", v. 97, №2, (2018), 188-192.

Основные возможности DMA-контроллера сопроцессора обработки сигналов и тесты для их проверки

А.А. Бурцев

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

Аннотация: В статье описываются основные возможности DMA-контроллера, используемого в микропроцессорах семейства КОМДИВ для передачи данных между основной памятью и локальной памятью сопроцессора обработки сигналов, которые способствуют ускоренному исполнению вычислительных программ.

Подробно рассматриваются приёмы формирования дескрипторов типа GET и PUT для выполнения типовых DMA-передач, а также объясняются ключевые аспекты реализации тестовой программы, разработанной для верификации одиночных DMA-передач.

Ключевые слова: SIMD-архитектура, микропроцессоры семейства КОМДИВ, сопроцессор обработки сигналов, DMA-контроллер.

Введение

В современных микропроцессорных системах команды вычислительных операций над данными в регистрах, как правило, исполняются значительно быстрее, чем команды, осуществляющие чтение обрабатываемых данных из оперативной памяти в регистры и запись их обратно из регистров в память.

Чтобы не тормозить вычислительный процесс и обеспечивать для него своевременную подкачку данных, подсистемы памяти, как правило, организуются в иерархическую структуру. В этой иерархии в качестве прослойки между основной памятью и регистрами используется быстродействующая накристальная память, в которой размещают данные, наиболее востребованные в текущий момент времени. Чаще всего такую накристальную память применяют в качестве кэша команд и данных 1-го или 2-го уровня. Но в некоторых приложениях (например, для задач реального времени) её могут использовать и как особую быстродействующую память.

Для ускоренного обмена данными между основной памятью и накристальной применяются каналы прямого доступа в память (ПДП или DMA). В таком случае подкачка новой группы данных может осуществляться параллельно с выполнением вычислительных действий над прежней группой данных. И вычислительная программа при этом не теряет производительность в ожидании поступления в быстродействующую память очередной порции обрабатываемых данных.

Реализуемые аппаратурой возможности каналов прямого доступа в память постепенно совершенствовались. И если ранее в микропроцессорных системах [1] DMA-каналы умели лишь перекачивать непрерывные блоки машинных слов из одного участка памяти в другой, то со временем [2,с.216] их наделили умением перебирать передаваемые слова памяти в произвольно заданном порядке как элементы многомерных массивов, а также выполнять над такими массивами передаваемых слов некоторые преобразования путём расстановки их элементов на новом месте в другом порядке.

В настоящее время усовершенствованные возможности DMA-каналов, позволяют, например, не просто передать матрицу из одной области память в другую, но при этом ещё и транспонировать её, что бывает требуется сделать для обеспечения её более ускоренной обработки при решении задач линейной алгебры [3].

Другой пример. При выполнении преобразования Фурье приходится выполнять над обрабатываемым вектором операцию так называемой бит-инверсной перестановки [4]. А ведь такую перестановку можно тоже совместить с передачей обрабатываемого вектора из одной области памяти в другую с помощью DMA-канала.

Таким образом, умело применяя новые усовершенствованные возможности DMA-

каналов, можно в результате повысить производительность программ, создаваемых для решения многих задач линейной алгебры и цифровой обработки сигналов.

Далее в статье предлагается более подробное знакомство с подобными «интеллектуальными» функциями контроллера DMA-канала, обеспечивающего передачу данных между глобальной памятью основного процессора и локальными блоками памяти его 4-х секционного SIMD-сопроцессора CP2, предназначенного служить ускорителем основных операций для задач цифровой обработки сигналов.

1. Сопроцессор CP2

В микропроцессорном семействе КОМДИВ [5] предусмотрены специализированные сопроцессоры (CPV и CP2) так называемой SIMD-архитектуры, которые позволяют совершать одну и ту же вычислительную команду сразу над несколькими комплектами данных, размещённых в их различных секциях. Такие сопроцессоры служат в качестве аппаратных ускорителей при решении многих задач, где требуется выполнять обработку больших массивов данных из вещественных и комплексных чисел.

В составе микропроцессоров BM6 и BM8 роль такого ускорителя исполняет так называемый векторный сопроцессор CPV. Он функционирует синхронно вместе с основным процессором в едином потоке команд. И ему напрямую доступна основная память (а также кэш-память), которую используют и основной процессор (CPU), и сопроцессор плавающей арифметики (CP1).

А в составе микропроцессоров BM7 и BM9 роль такого SIMD-ускорителя операций вещественной и комплексной арифметики исполняет сопроцессор сигнальной обработки CP2, который в отличие от CPV может функционировать параллельно с основным процессором и асинхронно по отношению к нему. Вычислительные возможности CP2 подробно охарактеризованы в [6].

CP2 содержит 4 вычислительных секции (ALU), каждая из которых имеет свою локальную память (LMem) для хранения обрабатываемых данных. Каждая вычислительная секция исполняет единую CP2-программу, но над данными своей секции. Передача данных из основной памяти (MEM) в локальную память каждой секции и обратно осуществляется с помощью контроллера прямого доступа DMA CP2 (см. рис.1).

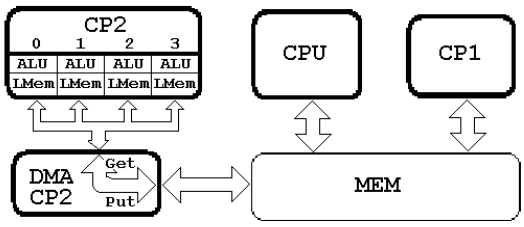


Рис. 1. Схема передачи данных между основной памятью и локальной памятью секций сопроцессора.

2. Порядок работы для запуска одиночной DMA-передачи

Передача данных из основной памяти в локальные памяти вычислительных секций сопроцессора CP2 и обратно осуществляется под управлением контроллера прямого доступа в память (ПДП) DMA CP2. Такой контроллер можно запрограммировать на выполнение сразу целой серии DMA-передач, для которых потребуется задать сложную канальную программу, предусматривающую перебор цепочки из нескольких дескрипторов. Но в данной статье будут рассматриваться только те возможности контроллера DMA CP2, которые ограничиваются режимом одиночной DMA-передачи.

В простейшем случае для одиночной DMA-передачи необходимо: 1) подготовить дескриптор, в котором задать параметры требуемой передачи; 2) поместить указатель на этот дескриптор в соответствующий регистр контролера (DSC_ADDR); и 3) взвести в регистре управления (CTRL) контроллера Start-бит (CTRL[0]) для запуска DMA-канала в работу.

После запуска DMA-передачи и до её завершения основной процессор (CPU) и сопроцессор (CP2) могут заниматься полезной вычислительной работой параллельно с осуществлением (под управлением контроллера) заказанной передачи данных. Но перед тем, как приступить к действиям над переданными DMA-каналом данными, потребуется проверить, что DMA-канал завершил работу, а если нет, то дождаться её окончания. Для этого следует убедиться, что в регистре состояния контроллера STATUS флаг Run (0-бит) оказался сброшенным (т.е. стал равен нулю).

Все параметры программируемой DMA-передачи задаются в её дескрипторе. В новой 64-разрядной версии контроллера каждый дескриптор Dscr состоит из 16-ти 64-разрядных слов. Этот блок из 16-ти слов можно рассматривать как массив Dscr[16] или

структуру с битовыми полями, в которых задаются разнообразные параметры программируемой передачи данных. Один из битов (31-ый в Dscr[0]) служит признаком действительности (валидации) дескриптора (DescrValid), а ещё в двух битах (25-26 Dscr[0]) задаётся тип дескриптора.

Передача данных из основной памяти MEM в память LMem секций CP2 задаётся дескриптором типа GET, а передача данных из локальной памяти секций CP2 в основную память – дескриптором типа PUT.

3. Дескриптор типа GET

Дескриптор GET предполагает формирование потока данных из 128-разрядных слов, прочитанных в определённом порядке из основной памяти (MEM), и передачу его по DMA-каналу в направлении от MEM к CP2 с записью передаваемых данных в локальные памяти секций CP2.

Все параметры дескриптора типа GET можно условно разделить на две группы. Первая MEM-группа параметров определяет, откуда и в каком порядке брать из основной памяти слова, из которых формируется передаваемый поток данных. А вторая CP2-группа параметров определяет, как и куда, по каким адресам и в каком порядке записывать в локальные памяти секций сопроцессора CP2 слова этого потока, поступающие из основной памяти.

3.1 GET-параметры MEM-группы

В MEM-группе параметров в полях GET-дескриптора задаются величины: Base, Bx, NX, NY, StepX, StepY. Роль этих параметров схематично поясняется с помощью рис. 2, на котором изображен двумерный массив элементов, размещённых в основной памяти, а стрелками задаётся порядок их перебора для взятия в формируемый поток данных (см. рис.3).

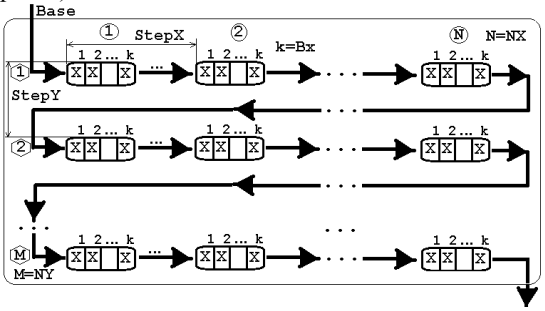


Рис. 2. Схема перебора слов основной памяти при формировании потока данных DMA-передачи GET.

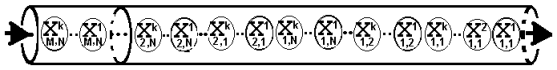


Рис. 3. Схематичное представление потока данных, сгенерированных при DMA-передаче типа GET.

Элементами такого воображаемого двумерного массива (матрицы) X[NY][NX] являются блоки, состоящие из нескольких 128-разрядных слов, расположенных в памяти непрерывно друг за другом. Количество слов в блоке задаётся параметром Bx дескриптора (**k** на рисунке). Минимальной порцией передаваемых данных является 128-разрядное слово (состоящее из 16 байт), так что можно считать, что элементами передаваемой матрицы являются непрерывные блоки из 16*Bx байтов.

Параметром NX дескриптора (**N** на рис.) задаётся количество элементов в одном ряду этой матрицы, а параметром NY (**M** на рис.) − количество рядов (строк) матрицы. Параметр Base определяет адрес расположения первого элемента матрицы в основной памяти. Заметим, что для корректной DMA-передачи требуется, чтобы этот адрес был кратен 8, т.е. указывал на начало 64-разрядного слова в памяти.

Элементы передаваемой матрицы не обязательно должны следовать в памяти непрерывно друг за другом. Следующий элемент ряда можно располагать в памяти с некоторым смещением от предыдущего. Это смещение (шаг) определяется параметром StepX, который задаёт значение, которое надо прибавить к адресу предыдущего элемента ряда, чтобы переместиться к следующему элементу того же ряда. (Уточним, что адресация в основной памяти осуществляется с точностью до байта).

Аналогичное смещение предусмотрено и для перехода от текущего ряда на следующий ряд матрицы. Оно определяется параметром StepY дескриптора и задаёт величину, которую надо прибавить к значению адреса первого элемента предыдущего ряда, чтобы получить адрес расположения первого элемента следующего ряда матрицы.

Таким образом, заданием параметров дескриптора GET этой группы можно, вообще говоря, выбрать из памяти элементы произвольной разреженной матрицы, чтобы сформировать из них поток данных, предназначенный для передачи в CP2. Заметим, что при необходимости формируемый поток данных можно также рассматривать и как 3-х мерный массив 128-разрядных слов XW[NY][NX][Bx] или байтов XB[NY][NX][16*Bx].

3.2 GET-параметры CP2-группы

Вторую CP2-группу GET-параметров можно условно тоже разделить на две части. В первой части задаются параметры, общие для всех секций сопроцессора CP2: cp2Base, cp2NX, cp2NY, cp2StepX, cp2StepY, W64Mode. А во второй части задаются параметры, индивидуальные для каждой **j**-ой секции: cp2BaseS**j**, cp2NXS**j**, cp2StepXS**j**, cp2StepYS**j** (**j**=0,1,2,3).

Параметры 1-ой части задают механизм генерации CP2-адресов, по которым поступающие из основной памяти 128-разрядные слова будут последовательно (в порядке их поступления) записываться в локальную память CP2-секций. Параметр cp2Base указывает начальный адрес локальной памяти, куда будет записываться первое слово поступающего потока.

Последующие слова потока будут записываться в локальную память как элементы двумерного массива. Параметр cp2NY задаёт, сколько будет рядов у этого массива, а параметр cp2NX задаёт, сколько элементов будет размещено в каждом ряду. Параметр cp2StepX определяет, на какую величину следует увеличить текущий адрес, чтобы перейти к следующему элементу ряда. А параметр cp2StepY – на сколько позиций (от первого элемента текущего ряда) следует увеличить адрес, чтобы перейти к первому элементу следующего ряда.

Можно сказать, что для генерации CP2-адресов применяется аналогичный механизм (схематично изображённый на рис. 2), который был использован для перебора блоков передаваемых слов основной памяти. Существенно, что для всех 4-х секций CP2 применяется единый механизм формирования адресов. Заметим также, что адресация в CP2 осуществляется с точностью до 64-разрядного слова.

В зависимости от флага-параметра W64Mode запись в секции по текущему сформированному адресу будет производиться 64-разрядными словами (если W64Mode=1) или парами таких слов (если W64Mode=0), представляющих младшую и старшую часть очередного 128-разрядного слова, поступающего из потока. Это необходимо учитывать, в частности, для корректного задания параметра cp2StepX: если поступающие из потока слова должны записываться в локальную память следом друг за другом как 64-разрядные слова, то данный параметр должен задавать в качестве адресного смещения значение 1, а если как 128-разрядные слова, то − значение 2.

Порядок разрешения записи в локальную память поступающих из потока слов можно регулировать индивидуально для каждой секции. Для этого в GET-дескрипторе для каждой секции по отдельности задаются 4 параметра: cp2BaseS**j**, cp2NXS**j**, cp2StepXS**j**, cp2StepYS**j** (**j**=0,1,2,3). Они то и определяют для каждой секции свой индивидуальный порядок перебора тех слов поступающего потока, которые должны быть записаны в её локальную память.

Поясним правило, определяющее, будет ли очередное слово потока записываться в **j**–ую секцию или нет. Для этого представим поступающий поток данных как последовательность 64-разрядных слов и пронумеруем все эти слова от 0 до P-1 (где P=Bx*NX*NY*2 – общее число передаваемых 64-разрядных слов). И пусть **T** – номер очередного слова, выбранного из потока. Вначале **T**=0, а далее каждый раз при взятии очередного слова из потока **T** увеличивается на 1 при режиме W64Mode=1 или на 2 при режиме W64Mode=0.

В каждой **j**-ой секции (**j**=0,1,2,3) ведётся аналогичный счётчик **Tj**, в котором последовательно перебираются номера тех элементов потока, которые должны быть записаны в локальную память **j**-ой секции. Вначале **Tj** равен значению параметра cp2BaseS**j**. Далее значение счётчика столько раз увеличивается на значение параметра cp2StepXS**j**, сколько итераций задано параметром cp2NXS**j**, а по исчерпании числа этих итераций, счётчик увеличивается на значение параметра cp2StepYS**j**, после чего приращения счётчика опять осуществляются cp2NXS**j** раз на величину cp2StepXS**j** и т.д.

Здесь следует особо подчеркнуть весьма важное обстоятельство, которое существенным образом регулирует представленный порядок перебора значений номеров для счётчика **Tj**. Оно заключается в том, что каждое последующее значение счётчик **Tj** в каждой секции (**j**=0,1,2,3) получает только после того, как будет произведена запись в локальную память **j**-ой секции того слова, номер которого **T** совпал с текущим значением **Tj**.

Выполняя DMA-передачу, заданную GET-дескриптором, контроллер DMA CP2 выбирает из сформированного потока очередное слово и сравнивает его номер **T** с текущими значениями счётчиков **T0**, **T1**, **T2**, **T3** каждой секции. Выбранное из потока слово записывается далее в локальную память каждой **j**-ой секции, счётчик которой **Tj** совпал

с номером **T** выбранного из потока слова. Другими словами, запись слова в **j**-ую секцию производится при условии: **Tj=T** (**j**=0,1,2,3). При этом важно отметить, что запись слова в локальную память осуществляется по единому для всех секций CP2-адресу, который был сгенерирован к данному моменту.

3.3 Примеры задания параметров для DMA-передач типа GET

Во всех приведённых далее примерах будем считать, что запись в локальную память всегда производится с начального адреса CP2_ADDR=0x800. Поэтому поле cp2Base GET-дескриптора задаём равным CP2_ADDR.

Сначала рассмотрим самый простейший случай. Пусть требуется организовать DMA-передачу из основной памяти в локальную память каждой секции (начиная с адреса CP2_ADDR) одномерного массива (вектора X[R]), состоящего из R 128-разрядных слов, непрерывно расположенных в памяти друг за другом. Далее полагаем, что параметр Base=X, т.е. всегда задаётся равным адресу начала расположения этого массива в основной памяти.

Для решения этой задачи можно предложить большое разнообразие вариантов задания параметров для подготовки GET-дескриптора. Рассмотрим несколько простейших вариантов для задания параметров MEM-группы:

```
A) Bx=R, NX=1, NY=1, StepX=…, StepY=…;
B) Bx=1, NX=R, NY=1, StepX=16, StepY=…;
C) Bx=1, NX=1, NY=R, StepX=…, StepY=16;
```

Заметим, что для поля Bx в Get-дескрипторе отводится 6 битов, значит, Bx можно задать от 0 до 63, поэтому вариант A возможен, только если R<64. Вообще говоря, представляя значение R в виде произведения трёх сомножителей: R=Rb*Rx*Ry, можно предложить множество других вариантов вида:

```
Bx=Rb, NX=Rx, NY=Ry,
StepX=16*Rb, StepY=16*Rb*Rx;
```

И для задания параметров CP2-группы в нашем распоряжении тоже имеется множество вариантов, каждый из которых получается в результате представления R в виде произведения двух сомножителей R=Rx*Ry. Причём такие варианты можно предложить как для режима записи 128-разрядных, так и для режима записи 64-разрядных слов:

```
W64Mode=0,  cp2NX=Rx, cp2NY=Ry,
cp2StepX=2, cp2StepY=2*Rx,
для j=0,1,2,3 { cp2BaseSj=0, cp2NXSj=Rx,
                cp2StepXSj=2, cp2StepYSj=2 } ;
```

```
W64Mode=1, cp2NX=2*Rx, cp2NY=Ry,
cp2StepX=1, cp2StepY=2*Rx,
для j=0,1,2,3 { cp2BaseSj=0, cp2NXSj=2*Rx,
                cp2StepXSj=1, cp2StepYSj=1} ;
```

Во всех этих случаях непрерывный блок 128-разрядных слов передаётся из основной памяти в CP2 и записывается в локальную память каждой его секции. А если такой блок при DMA-передаче потребуется записывать не во все, а лишь в отдельные секции CP2, то как в таком случае следует изменить параметры GET-дескриптора?

Чтобы не производить запись в **j**-ую секцию, достаточно в таком случае задать параметр cp2BaseS**j** этой секции равным общему числу P передаваемых 64-разрядных слов (P=2*R). Ведь это гарантирует, что текущий счётчик номеров **Tj** этой секции никогда не совпадёт с текущим значением счётчика **T**, который будет принимать значения в диапазоне от 0 до P-1.

Теперь допустим, что в **j**-ую секцию надо записывать не весь передаваемый массив 64-разрядных слов (в режиме W64Mode=1), а только какой-то его отдельный участок, например, отрезок длиной L слов, начиная с K-го. Тогда можно задать индивидуальную группу параметров для **j**-ой секции следующим образом:

```
cp2BaseSj=K, cp2NXSj=L,
cp2StepXSj=1, cp2StepYSj=P
```

Допустим, передаваемый поток представляет 64-разрядные элементы матрицы X[4][N] (P=4*N и N чётно) в порядке их следования «по строкам». И требуется передать его в CP2 так, чтобы в локальную память **j**-ой секции была записана **j**-ая строка матрицы. Для осуществления этой передачи можно предложить GET-дескриптор с такими параметрами:

```
Bx=1, NX=N/2, NY=4, StepX=16, StepY=8*N;
W64Mode=1, cp2NX=N, cp2NY=4,
cp2StepX=1, cp2StepY=0,
для j=0,1,2,3 { cp2BaseSj=j*N, cp2NXSj=N,
                cp2StepXSj=1, cp2StepYSj=P } ;
```

Заметим, что в нём параметр cp2StepY задан нулевым, чтобы отсчёт CP2-адресов для очередных N слов (очередной строки матрицы) начинался с начального адреса, заданного параметром cp2Base.

Если же передаваемый поток представляет собой матрицу X[N][4], и требуется его передать в CP2 так, чтобы в локальную память **j**-ой секции записать **j**-ый столбец этой матрицы, то для такой DMA-передачи можно предложить GET-дескриптор с вот такими параметрами:

```
Bx=1, NX=2, NY=N, StepX=16, StepY=32;
W64Mode=1, cp2NX=4, cp2NY=N,
 cp2StepX=0, cp2StepY=1,
 для j=0,1,2,3 { cp2BaseSj=j, cp2NXSj=1,
                 cp2StepXSj=…, cp2StepYSj=4 } ;
```

Заметим, что в этом дескрипторе уже параметр cp2StepX задан нулевым, чтобы для очередных 4-х слов, которые как очередные элементы столбцов надо будет последовательно поместить в 0-ую, 1-ую, 2-ую и 3-ью секции, CP2-адрес оставался одинаковым.

Теперь представим передаваемый из основной памяти поток из P 64-разрядных слов как последовательность элементов матрицы X[M][N], выбранных из неё в порядке их следования «по строкам» (полагаем, что N чётно и M*N=P). И потребуем записать эту последовательность элементов в каждую секцию CP2 как транспонированную матрицу Y[N][M], т.е. разместить их в локальной памяти секций в порядке «по столбцам».

Для решения этой задачи можно предложить следующий вариант задания параметров для GET-дескриптора:

```
Bx=1, NX=N/2, NY=M, StepX=16, StepY=8*N;
W64Mode=1, cp2NX=M, cp2NY=N,
 cp2StepX=M, cp2StepY=1,
 для j=0,1,2,3 { cp2BaseSj=0, cp2NXSj=M*N,
                 cp2StepXSj=1, cp2StepYSj=…} ;
```

Заметим, что в нём значение cp2StepYS**j** для **j**=0,1,2,3 можно не задавать, т.к. первой итерацией в каждой секции (по cp2NXS**j)** уже будут перебраны номера всех передаваемых слов (P=M*N).

Немного усложним задачу. Пусть передаётся трёхмерный массив 64-разрядных слов X[4][M][N], каждая из 4-х плоскостей которого представляет собой матрицу M*N (P=4*M*N). И требуется одной DMA-передачей разложить все его 4 плоскости в 4 секции CP2 так, чтобы каждая его **j**-ая плоскость (**j**=0,1,2,3) была положена в локальную память **j**-ой секции как транспонированная матрица Y[N][M].

Для решения такой усложнённой задачи можно предложить сформировать GET-дескриптор со следующими параметрами:

```
 Bx=1, NX=N*M/2, NY=4, StepX=16, StepY=8*N*M;
W64Mode=1, cp2NX=M, cp2NY=N,
 cp2StepX=M, cp2StepY=1,
для j=0,1,2,3 {
   cp2BaseSj=j*N*M, cp2NXSj=N*M,
   cp2StepXSj=1, cp2StepYSj=4* N*M } ;
```

Заметим, что в нём индивидуальные параметры каждой **j**-ой секции заданы так, чтобы осуществлять в этой секции перебор номеров только слов **j**-ой плоскости передаваемого потока. А для завершения перебора в **j**-ой секции (когда исчерпается итерация из N*M номеров, начатая с номера **j***N*M) применяется уже рассмотренный выше приём назначения параметру cp2StepYS**j** (**j**=0,1,2,3) значения общего количества передаваемых слов P=4*N*M.

В рассмотренных выше примерах индивидуальные параметры секций в GET-дескрипторе задавались по некоторому единому правилу в виде общих формул, зависящих только от номера секции **j**. Теперь попробуем решить такую задачу, когда требуется в локальные памяти разных секций передавать разные по форме части одной и той же матрицы.

Пусть в DMA-потоке передаётся всё та же матрица X[M][N] 64-разрядных слов, P=M*N. И требуется сформировать GET-дескриптор для такой DMA-передачи, чтобы в память 0-й секции скопировать такую матрицу всю целиком (M*N слов), в 1-ую секцию отправить её **L**-ую строку (N слов), во 2-ую секцию поместить её **K**-ый столбец (M слов), а в 3-ью секцию записать её подматрицу X[L:M][K:N], т.е. своеобразную «прямоугольную вырезку» из неё, которая представляет собой матрицу, в левом верхнем углу которой стоит элемент X[**L**][**K**], а в нижнем правом – последний элемент основной матрицы X[M-1][N-1].

Предложим вариант GET-дескриптора для осуществления такой DMA-передачи:

```
Bx=1, NX=M*N/2, NY=1, StepX=16, StepY=8*M*N;
W64Mode=1, cp2NX=M*N, cp2NY=1,
 cp2StepX=1, cp2StepY=0,
   cp2BaseS0=0,  cp2NXS0=M*N,
 cp2StepXS0=1, cp2StepYS0=P,
   cp2BaseS1=L*N,  cp2NXS1=N,
 cp2StepXS1=1, cp2StepYS1=P,
   cp2BaseS2=K,  cp2NXS2=M,
 cp2StepXS2=N, cp2StepYS2=P,
   cp2BaseS3=L*N+K,  cp2NXS3=N-K,
 cp2StepXS3=1, cp2StepYS3=K+1 ;
```

Заметим, однако, что этим дескриптором удаётся решить поставленную задачу с весьма существенной оговоркой: передаваемый в каждую секцию фрагмент матрицы не всегда будет записываться с начального адреса, заданного в cp2Base, а помещаться в соответствующее ему место такой же матрицы в предположении, что она расположена, начиная с адреса cp2Base.

До сих пор в примерах DMA-передач для формирования потока использовался двумерный или трёхмерный массив слов, расположенных в основной памяти непрерывно друг за другом. Однако, в более общем случае может потребоваться передавать массив,

элементы которого следуют в памяти с некоторым интервалом (шагом).

Например, такая потребность возникает в случае, когда для последующей обработки на СР2 надо передавать не всю матрицу , а какую-то её часть, которая тоже может считаться матрицей (двумерным массивом), как будто бы вырезанной из основной. Элементы такой матрицы располагаются в основной матрице не обязательно подряд друг за другом, а с некоторым шагом. Именно такая разновидность матрицы демонстрируется в общем виде на рис. 2.

Допустим, объявлена и размещена в памяти матрица 128-разрядных слов W[U][V]. И из её отдельных элементов собрана матрица X[M][N] по такому правилу:

```
X[m][n]= W[m*q+f][n*p+g]
m=0,1,..,M-1; n=0,1,..,N-1; где q,f,p,g – константы
```

Строки матрицы X формируются из строк матрицы W с номерами вида m*q+f, а столбцы − соответственно из столбцов матрицы W с номерами вида n*p+g. Так что первый элемент матрицы X[0][0]=W[f][g], соседние элементы её строки располагаются в матрице W на расстоянии p элементов, а соседи по столбцу − на расстоянии q элементов в матрице W.

Чтобы обеспечить DMA-передачу такой матрицы X в СР2, потребуется задавать такие параметры MEM-группы GET-дескриптора:

```
Base=&W[f][g], Bx=1, NX=N, NY=M,
StepX= 16*p,      StepY= q*16*p*N;
```

4. Дескриптор типа PUT

Дескриптор PUT предназначен для передачи в основную память (MEM) потока 128-разрядных слов, прочитанных из локальных памятей (LMem) секций сопроцессора СР2. Задаваемые в его полях параметры можно тоже условно разделить на группы MEM- и CP2-.

MEM-группу составляют те же параметры (Base, Bx, NX, NY, StepX, StepY), которые уже рассматривались в п.3.1 и пояснялись на рис. 2. Но в дескрипторе PUT они задают последовательность адресов основной памяти, куда будут записываться 128-разрядные слова, передаваемые из секций СР2.

CP2-группу параметров дескриптора PUT можно также представить состоящей из двух частей. В первой её части задаются параметры cp2Base, cp2NX, cp2NY, cp2StepX, cp2StepY, отвечающие за генерацию последовательности CP2-адресов, по которым из локальной памяти (LMem) каждой секции будут выгружаться 128-разрядные слова для передачи их по DMA-каналу в основную память (MEM). Эти параметры в дескрипторе PUT применяются для генерации CP2-адресов по такому же правилу, которое было пояснено в п. 3.2 для режима передачи 128-разрядных слов.

Вторую же часть CP2-группы в упрощённом виде можно представить состоящей лишь из двух параметров PutFirst и PutFlow, которые задают, в каком порядке слова, выбираемые из 4-х различных секций CP2, должны следовать в передаваемом DMA-потоке для записи их в основную память.

Параметр PutFlow определяет, по сколько 128-разрядных слов следует выгружать в поток из каждой секции. Ненулевое значение p этого параметра задаёт такой порядок передачи данных из секций CP2: сначала p слов из 0-й секции, затем p слов из 1-й секции, потом p слов из 2-й секции и следом p слов из 3-й секции, а далее снова по p слов из 0-й секции, из 1-й секции, из 2-й, из 3-ей и т.д. пока не закончится перебор всех слов, предназначенных к DMA-передаче в каждой из 4-х секций.

При нулевом значении параметра PutFlow из каждой секции выгружаются сразу все слова, предназначенные для DMA-передачи. И тут решающую роль играет параметр PutFirst: он то и определяет, из какой секции начнётся выгрузка слов. Скажем, если PutFirst=s, то первой выгружается секция с номером s (s=0..3), после неё выгружается секция с номером (s+1)mod 4 и т.д. по цепочке номеров (0-1-2-3) пока не будут выгружены слова из всех 4-х секций CP2.

Если из каждой секции выгружается H слов, то общее число слов, передаваемых из CP2 в основную память, будет равно Q=4*H. Это необходимо учитывать при задании параметров MEM-группы: желательно, чтобы число 128-разрядных слов (Q), предназначенных для передачи из всех секций CP2, совпадало с количеством адресов (P) для записи в основную память, которые будут сгенерированы в зависимости от установленных параметров MEM-группы (P=Bx*NX*NY).

Поскольку PUT-дескриптором предопределяется DMA-передача данных из каждой секции CP2, возникает естественный вопрос: а можно ли в основную память передать данные не из всех секций, а, скажем, только из одной? И как это осуществить?

Ответ будет такой: данные из каждой секции всё равно будут выгружаться и передаваться для записи в указанную область основной памяти, а чтобы в заданном участке памяти оказались в итоге данные из требуемой

секции, следует эту секцию выгружать последней и начинать запись данных очередной секции в основную память с одного и того же адреса.

Пусть, например, требуется в массив X[R], размещённый в основной памяти, передать блок из R 128-разрядных слов **s**-ой секции, расположенный в её локальной памяти, начиная с адреса CP2_ADDR=0x800. Для этой DMA-передачи можно предложить создать PUT-дескриптор со следующими параметрами:

```
Base=X, Bx=1, NX=R, NY=4, StepX=16, StepY=0,
cp2Base= CP2_ADDR, cp2NX=R, cp2NY=1,
cp2StepX=2, cp2StepY=…,
PutFlow=0, PutFirst=(s+1)&0x3; // =(s+1)mod 4;
```

Согласно этому дескриптору из каждой секции будет передаваться R 128-разрядных слов. Нулевое значение параметра StepY гарантирует, что запись в массив X будет осуществляться с его начального адреса через каждые R слов передаваемого потока. А поскольку выгружаться первой будет секция с номером (**s**+1)mod 4, то последней в массив X будет загружена секция с номером **s**, а значит, в нём в итоге окажутся R слов именно из этой секции.

Одной DMA-передачей с последовательной выгрузкой слов из секций можно, например, уложить данные в двумерный массив X[4][N] так, чтобы в каждую его **s**-ую строку записался вектор из N слов, переданный из **s**-ой секции:

```
Base=X, cp2Base= CP2_ADDR,
Bx=1, NX=N, NY=4, StepX=16, StepY=16*N,
cp2NX=N, cp2NY=1, cp2StepX=2, cp2StepY=…,
PutFlow=0, PutFirst=0;
```

А возможностью выгрузки слов с чередованием секций можно, например, воспользоваться, чтобы одной DMA-передачей уложить данные в двумерный массив X[N][4] так, чтобы в каждый его **s**-ый столбец записался вектор из N слов, переданный из **s**-ой секции:

```
Base=X, cp2Base= CP2_ADDR,
Bx=1, NX=4, NY=N, StepX=16, StepY=16*4,
cp2NX=N, cp2NY=1, cp2StepX=2, cp2StepY=…,
PutFlow=1, PutFirst=…;
```

В качестве примера демонстрации более продвинутых возможностей PUT-дескриптора, покажем, как можно решить своеобразную обратную задачу по отношению к той усложнённой задаче, которая предлагалась в п.3.3.

Пусть требуется (одной DMA-передачей!) в трёхмерный массив 128-разрядных слов X[4][M][N] основной памяти передать из локальной памяти каждой **s**-ой секции CP2 (**s**=0,1,2,3) матрицу Y[N][M] так, чтобы она разместилась в **s**-ой плоскости массива X в транспонированном виде как матрица M*N.

Полагая, что в каждой секции матрица Y расположена, начиная с адреса CP2_ADDR, можно предложить для решения этой задачи PUT-дескриптор с такими параметрами:

```
Base=X, cp2Base= CP2_ADDR,
Bx=1, NX=M*N, NY=4,
StepX=16, StepY=16*M*N,
cp2NX=N, cp2NY=M, cp2StepX=2*M, cp2StepY=2,
PutFlow=0, PutFirst=0;
```

5. Программа тестирования одиночных DMA-передач

С целью проверки правильного исполнения описанных функциональных возможностей контроллера DMA по передаче данных между основной памятью и блоками локальной памяти секций CP2 для среды KmdTestKit [7], используемой в ФНЦ НИИСИ РАН для тестирования логических моделей, автором была разработана специальная тестовая программа (CP2_DMA_Test). Скомпонованная в среде KmdTestKit для платформы DSP64 она запускалась на Си-модели (vmips) и на RTL-модели микропроцессора BM9 (а также на его реализации в ПЛИС'е) на этапе верификации новой 64-разрядной версии контроллера DMA CP2.

Тестовая программа состоит из нескольких независимых частей. Каждая из них запускает несколько тестов DMA-передач. В совокупности эти тесты охватывают все примеры DMA-передач, рассмотренные в данной статье, прогоняя их многократно с различными значениями настраиваемых параметров.

В каждом тесте DMA-передачи типа GET область локальной памяти секций, куда будет производиться запись, предварительно очищается (прописывается особыми «бланковыми» словами). После выполнения передачи данные из локальной памяти прочитываются в основную память и сравниваются с эталонными данными, переданным по каналу DMA.

Аналогично при DMA-передаче типа PUT предварительно очищается область основной памяти, куда будет производиться запись, а эталонные данные, предназначенные для DMA-передачи, прописываются в локальные памяти секций.

Такие вспомогательные операции чтения блоков слов из локальной памяти секций и записи в неё в этой тестовой программе осуществляются с помощью функций

специальной библиотеки (k128cp2), имеющейся в среде KmdTestKit, которая обеспечивает доступ к регистрам сопроцессора СР2 и словам локальной памяти его секций через особые коммуникационные регистры.

При разработке тестовой программы приходилось учитывать некоторые особенности функционирования DMA-канала. Дело в том, что и дескриптор, и передаваемые данные контроллер DMA всегда читает из основной памяти (и записывает в неё), минуя кэш-память. И чтобы быть уверенным, что контроллер работает с обновлённой версией данных, перед запуском DMA-передачи надо гарантировать, что данные не остались в кэше, а успели быть записаны в память. А после DMA-передачи надо соответственно обращаться к переданным данным, минуя кэш, напрямую в основную память, где они были обновлены только что выполненной DMA-передачей.

Чтобы решить такую проблему обеспечения достоверности (когерентности) данных, можно очищать каждый раз кэш до, а затем и после DMA-передачи. Но в тестовой программе CP2_DMA_Test было применено другое решение этой проблемы: адреса всех дескрипторов и массивов передаваемых данных используются с взведённым 15-ым битом, а это предписывает (в микропроцессорах КОМДИВ) сквозное обращение напрямую в память.

Ещё с одной интересной проблемой синхронизации пришлось столкнуться при отладке совместной работы тестовой программы и контроллера DMA СР2. Дело в том, что для заполнения параметров дескриптора используются команды записи в память, а для запуска DMA-канала в работу используются команды записи в регистры контроллера. При конвейерном исполнении этих групп команд может оказаться, что команды записи в память, взятые на исполнение раньше команд записи в регистры, могут ещё не успеть произвести реальную запись в ячейки физической памяти, а DMA-канал уже может начать работу и приступить к чтению ещё не до конца заполненного дескриптора.

Чтобы этого не допустить, перед началом заполнения дескриптора сбрасывается бит его валидации (DescrValid), который взводится после заполнения всех полей дескриптора. А непосредственно перед запуском DMA-канала в работу (т.е. перед установкой Start-бита в регистре управления CTRL), выполняется цикл ожидания, который периодически считывает напрямую из памяти 0-е слово дескриптора, чтобы убедиться, что бит валидации дескриптора (DescrValid) в нём уже установлен.

Заключение

Конечно, описанные в статье функции контроллера DMA СР2 не раскрывают полностью все его возможности. Подробно охарактеризованы лишь те его функции, которые позволяют выполнять одиночные DMA-передачи. Не были затронуты вниманием дескрипторы типа JUMP, интерфейсные регистры, флаги синхронизации DMA-передачи с работой сопроцессора СР2, маскирование передаваемых данных, а также бит-реверсный порядок их расстановки.

Однако, умелое использование даже такого минимального ассортимента возможностей DMA-контроллера на практике уже позволяет реально ускорить многие вычислительные программы путём совмещения рутинных операций по передаче данных из одной области памяти в другую параллельно с выполнением высокопроизводительных вычислительных операций.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований по теме «Разработка архитектуры, системных решений и методов для создания микропроцессорных ядер и коммуникационных средств семейства систем на кристалле двойного назначения» (№0065-2019-0004).

Main capabilities of the DMA controller of the signal processing coprocessor and tests for its verification

A.A.Burtsev

Abstract. In article the main capabilities of the DMA controller used in microprocessors of the KOMDIV family for data transfers between a base memory and a local memory of the signal processing coprocessor which promote the accelerated execution of computing programs are described.

In detail methods of forming of GET and PUT descriptors for execution of standard DMA transfers are considered and also key aspects of implementation of the test program developed for verification of single DMA transfers are explained.

Keywords: SIMD architecture, microprocessors of the KOMDIV family, signal processing coprocessor, DMA controller.

Литература

1. Дж. Уокерли. Архитектура и программирование микро-ЭВМ: в 2-х книгах. М.: Мир, 1984. кн. 1, 418 − 423.

2. В.В. Корнеев, А.В. Киселёв. Современные микропроцессоры. М.: Нолидж, 2000.

3. А.А. Бурцев. О возможности оптимизации некоторых функций библиотеки линейной алгебры с помощью векторного сопроцессора. «Труды НИИСИ РАН», т. 4 (2014), №2, 5 − 15.

4. А.А. Бурцев. О возможности применения векторного сопроцессора для ускорения операции быстрого преобразования Фурье «Труды НИИСИ РАН», т. 5 (2015), №2, 138 − 147.

5. Микропроцессоры семейства КОМДИВ. URL: https://www.niisi.ru/devel.htm (31.05.2019).

6. О.Ю. Сударева. Эффективная реализация алгоритмов быстрого преобразования Фурье и свёртки на микропроцессоре КОМДИВ128-РИО. М.: НИИСИ РАН, 2014. 119 − 139.

7. С.А. Сидоров, А.Б. Слепов. Система тестирования логических моделей KMDTESTKIT. «Труды НИИСИ РАН», т. 8 (2018), №1, 37 − 43.

Сканирование сети RapidIO в самоконтролируемых программах параллельной обработки сигналов для мультипроцессорных комплексов реального времени

Т.К. Грингауз[1], А.Н. Онин [2]

ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН»,
Отдел математического обеспечения, Москва, Россия,
E-mail's: [1]gring@niisi.ras.ru, [2]alexii@niisi.ras.ru

Аннотация: Рассматриваются мультипроцессорные комплексы реального времени с коммуникационной средой RapidIO [1]. В целях самоконтроля мультипроцессорных приложений результаты локального хронометража программ собираются на аккумулирующем процессоре в цикле вычислительного конвейера[2]. Для большей эффективности самоконтроля предлагается дополнить технологию [2] периодическим опросом регистров коммутаторов RapidIO. Описывается метод сканирования сети RapidIO.

Ключевые слова: Самоконтроль системы, библиотека параллельной обработки сигналов, библиотека сбора данных самоконтроля, стадия-аккумулятор, функция-обработчик, коммуникационная среда RapidIO, оконечное устройство, коммутатор.

1. Введение

Статья продолжает работу [2], посвященную технологии сбора данных самоконтроля для программ параллельной обработки сигналов в реальном времени. Объект самоконтроля - мультипроцессорная целевая программа с архитектурой БПОС [3] в среде операционной системы реального времени семейства ОС Багет 3.x [4] (далее – ОСРВ). Аппаратная платформа - процессоры архитектуры КОМДИВ (1890ВМ6Я, 1890ВМ7Я, 1890ВМ8Я), коммуникационная среда RapidIO [1]. Состав средств общего программного обеспечения и особенности их применения описаны в [5]-[8].

Программа с архитектурой БПОС осуществляет конвейерную обработку входного потока данных. Программа представляет собой последовательность вычисли-тельных стадий, обменивающихся потоками данных. Конвейер тактируется входными данными, поступающими с фиксированной частотой. Ошибки проектирования, ошибки программы или аппаратуры могут проявляться как зависание вычислительного конвейера или как замедление его работы. Конвейер работает со скоростью самого медленного узла. При замедлении одного узла замедляются и все остальные. Задача самоконтроля – локализовать источник замедления. Можно условно выделить два уровня локализации источника замедления: 1) локализация с точностью до объекта БПОС [3] (вычислительная стадия, группа процессоров, логический номер процессора, поток данных), 2) локализация с точностью до физического узла RapidIO [1] (оконечное устройство, коммутатор).

Для самоконтроля системы можно использовать две категории данных. К первой относятся данные, определенные как программные объекты и сохраняемые в оперативной памяти (показания таймеров, значения переменных, и т.д.). Такие данные порождаются на процессорах, выполняющих программу. Вторая категория - это данные, записываемые аппаратурой (процессорами и микроконтроллерами) в специальные регистры (регистры статуса, регистры управления ошибками, и т.д.). Такие данные порождаются всеми узлами RapidIO (как процессорными элементами, так и коммутаторами). Программа может обращаться без выхода в сеть к программным объектам и к регистрам процессора (далее – «локальные данные»). Опрос регистров коммутаторов возможен только путем отправки в

них служебных пакетов типа MAINTENANCE по RapidIO с удаленного процессора. Программа [2] предназначена для сбора локальных данных самоконтроля. Эти данные передаются на аккумулирующий процессор, где выполняется их анализ.

Технологию [2] проверили на примере модельной задачи с искусственно созданным замедлением коммутатора RapidIO. Целью была локализация источника замедления с точностью до потока данных [3], проходившего через «сбойный» коммутатор. Замедлившийся поток идентифицировали путем анализа локальных данных самоконтроля. Проверка показала, что для локализации замедлений, вызванных коммутатором, таких данных не достаточно. Для достижения точного результата потребовалось анализировать содержимое регистров коммутаторов RapidIO.

С целью парирования проблемы программа сбора данных самоконтроля [2] доработана. В нее включены два вида опроса коммутаторов RapidIO. Первый – сканирование маршрута передачи данных между двумя процессорами в случае снижения скорости передачи ниже порогового уровня. Второй – периодический опрос регистров коммутаторов на заданной частоте. В статье описаны метод сканирования сети RapidIO и его интеграция с технологией [2].

2. Справка по архитектуре и принципу функционирования программы сбора данных самоконтроля

В состав программы сбора данных самоконтроля [2] входят:

- библиотека интерфейсных функций,
- стадия-аккумулятор.

Интерфейсные функции вызываются из БПОС, из стадии-аккумулятора, из прикладной программы.

Программа предназначена для сбора системных и пользовательских данных самоконтроля. Системные данные определены в БПОС. Структура и семантика пользовательских данных определяется разработчиком прикладной программы. Данные самоконтроля порождаются на локальных процессорах и с помощью интерфейсных функций передаются на аккумулирующий процессор на каждой итерации конвейерного цикла. На аккумулирующем процессоре выполняется стадия-аккумулятор. Она осуществляет:

- сбор данных от локальных процессоров и их семантический разбор,
- сохранение данных самоконтроля в структуре метаданных по принципу кольцевого буфера,
- вызов пользовательских обработчиков предопределенных событий,
- доступ к метаданным из обработчиков событий посредством интерфейсных функций.

Транспортный механизм передачи данных самоконтроля на аккумулирующий процессор - транзакции типа NWRITE в среде RapidIO.

Тракт данных самоконтроля изображен на рис.1.

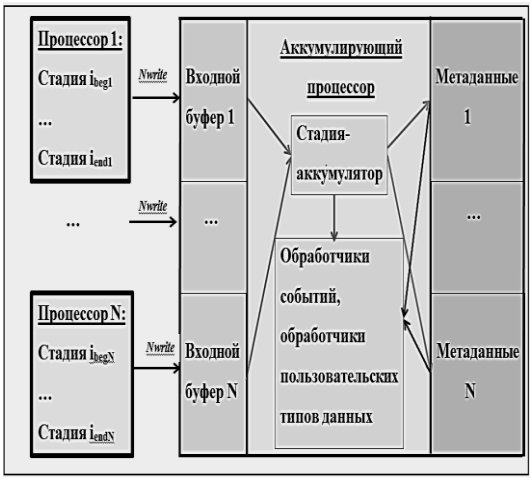


Рис.1 Тракт данных самоконтроля

3. Результаты эксперимента. Необходимость мониторинга коммутаторов

Программа сбора данных самоконтроля [2] фиксирует медленную передачу данных по RapidIO следующим образом.

Данные передаются потоками БПОС по каналам операционной системы (далее – «mp-канал»). Каждый mp-канал осуществляет передачу типа «точка-точка». [2]. С mp-каналом связаны входной и выходной порты потока БПОС. Порту соответствует очередь буферов [2,3]. Критерий медленной передачи данных – одновременное выполнение двух условий: 1) на стороне процессора-отправителя заняты все выходные буфера, 2) на стороне процессора-приемника свободны все входные буфера [2].

В процессе функционирования БПОС записывает текущее количество свободных буферов на стороне источника и приемника в структуры портов. Эта информация передается на аккумулирующий процессор в составе системных данных самоконтроля [2]. Стадия-аккумулятор анализирует данные и при выявлении медленной передачи вырабатывает событие
STATISTIC_EVENT_TRANSFER_SLOWLY.

Функция-обработчик этого события выводит в консоль аккумулирующего процессора сообщение, по которому можно идентифицировать процессор-источник, номер итерации конвейера, время события, mp-канал, количество свободных буферов на источнике и на приемнике [2].

Мы провели эксперимент по выявлению источника замедления модельной задачи. Источник замедления определялся по первому появлению на аккумулирующем процессоре события
STATISTIC_EVENT_TRANSFER_SLOWLY.
Эксперимент описан ниже.

На рис. 2 изображен фрагмент структуры RapidIO

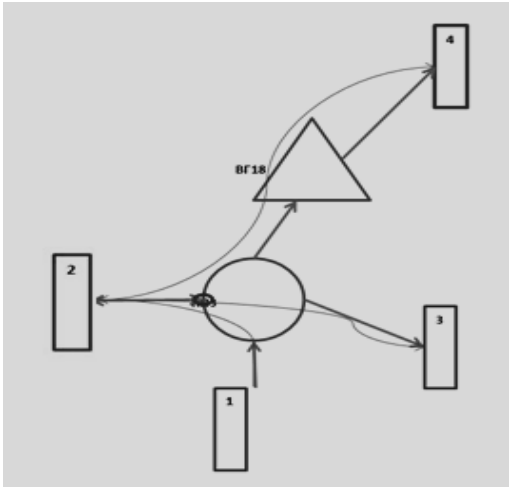


Рис. 2. Пример ошибочной локализации источника замедления.

Условные обозначения:

- прямоугольники с номерами 1,2,3,4 – процессоры КОМДИВ,
- окружность – коммутатор PRIO 1890КП3Я (далее - «КП3»),
- треугольник – коммутатор PRIO/SRIO 1890ВГ18Я (далее - «ВГ18»),
- прямые стрелки – соединения портов устройств RapidIO. Стрелки ориентированы в направлении потоков данных задачи,
- кривые линии – три потока данных: 1->2, 2->3, 2->4.

Микросхема 1890ВГ18Я поддерживает два режима SRIO: 1x, 4x [1]. Штатное функционирование модельной программы рассчитано на режим 4x. В эксперименте использовался режим 1x. В этом режиме данные передаются в 4 раза медленнее. Таким образом, тракт данных потока 2->4 проходил через источник замедления - микросхему 1890ВГ18Я.

Ожидаемый результат: мы предполагали сначала зафиксировать замедление потока 2->4, а затем - замедление потока 1->2 (поскольку поток 1->2 ближе к началу конвейера, чем 2->4).

На практике стадия-аккумулятор зафиксировала замедление потока 1->2, а сообщение о замедлении потока 2->4 не появилось. Наблюдалось следующее состояние входных и выходных портов потоков:

-на процессоре 1 (порт потока 1->2) заняты все выходные буфера,

-на процессоре 2 (порт потока 1->2) свободны входные буфера,

-на процессоре 2 (порт потока 2->3) свободны выходные буфера,

-на процессоре 2 (порт потока 2->4) свободны выходные буфера,

-на процессоре 3 (порт потока 2->3) есть свободные входные буфера,

-на процессоре 4 (порт потока 2->4) есть свободные входные буфера.

Приведенное распределение буферов показывает, что замедлился поток 1->2, а потоки 2->3, 2->4 передают данные без задержки.

Причина ошибочной локализации источника замедления заключается в следующем.

Тракт потока 1->2 проходит через коммутатор 1890КП3Я, тракт потока 2->3 – через коммутатор 1890ВГ18Я. С портом коммутатора RapidIO связана очередь входных буферов. Передача данных в коммутатор возможна лишь в том случае, когда во входной очереди порта есть свободный буфер. Механизм передачи сообщений по mp-каналу синхронный. Источник отправляет приемнику запрос на передачу данных. Приемник отправляет положительный или отрицательный ответ. Источник отправляет данные только по получении положительного ответа.

Восстановим последовательность событий в рассматриваемом примере.

В связи с замедлением функционирования микросхемы 1890ВГ18Я данные потока 2->4

заняли все буфера входного порта коммутатора 1890КП3Я (на рис.2 порт обведен жирной окружностью). В результате поток 1->2 стал передавать данные на пониженной частоте. Вот как это происходит. Процессор 1 отправляет запрос в 2 на передачу данных. Процессор 2 отправляет положительный ответ. Отправка ответа в процессор 1 задерживается, поскольку все буфера порта 1890КП3Я заняты. Поток 1->2 замедляется. Мы видим характерную картину его замедления: на источнике буфера заняты, на приемнике – свободны. Для последующих участков конвейера (потоки 2->3, 2->4) поток 1->2 – входной. Вся следующая часть конвейера успевает работать на частоте потока 1->2, поэтому характерной картины замедления потоков 2->3, 2->4 мы не наблюдаем.

Приведенный пример показывает, что для точной локализации источника замедления не достаточно анализировать очереди буферов входных и выходных портов. Необходимо использовать информацию о состоянии коммутаторов RapidIO.

В программу сбора данных самоконтроля добавлен опрос коммутаторов RapidIO. Описание технологии приведено в последующих разделах. Доработанную программу применили в рассматриваемом примере. Результат оказался правильным: стадия-аккумулятор сначала зафиксировала переход микросхемы 1890ВГ18Я в режим 1x, а затем замедление потока 1->2.

4 Технология опроса коммутаторов RapidIO

4.1 Виртуальный образ сети RapidIO. Виртуальное сканирование

Ниже описаны доработки, внесенные в программу сбора данных самоконтроля [2].

Под виртуальным образом сети RapidIO (далее – «виртуальный образ») мы понимаем набор структурированных данных об узлах RapidIO, об их портах, соединениях, об установленных в сети маршрутах. Данные об узле включают его тип, РИО-идентификатор и другие метаданные (например, логическое имя или принадлежность той или иной подсистеме комплекса). Каждому узлу RapidIO в виртуальном образе соответствует уникальный идентификатор (далее - «виртуальный идентификатор»). Виртуальный идентификатор однозначно определяет физическое местоположение узла. По виртуальному идентификатору можно извлекать все данные об узле, хранящиеся в виртуальном образе.

Примечание. Логическое имя – это символическое имя, составленное в соответствии с правилами БПОС по номеру группы и номеру процессора в группе [3]. По логическому имени однозначно определяется логический номер процессора [3].

В программе сбора данных самоконтроля виртуальный образ реализован как набор структур, заполняемых на этапе инициализации. Исходные данные для заполнения структур содержатся в «файле статической инициализации RapidIO» формата BMRW (далее - «файл bmrw»). Этот файл создается на инструментальной ЭВМ под управлением ОС Linux на этапе подготовки задачи к запуску [6]. Файл bmrw формируется посредством ПИ УПЗ-РИО [6] на основе формализованного описания аппаратной конфигурации комплекса и конфигурации задачи БПОС [6]. Для доступа программы сбора данных самоконтроля к файлу bmrw необходимо включить его в образ операционной системы в виде tar-файла.

Для того, чтобы опросить коммутатор RapidIO с процессора (далее - «процессор-источник»), необходимо отправить в него пакет типа MAINTENANCE [1]. Это возможно в том случае, если через коммутатор проходит маршрут с процессора-источника в какое-либо оконечное устройство (приемник). Для адресации пакета нужно указать РИО-идентификатор приемника и число шагов до коммутатора, обозначаемое «hop» [1] (далее - «число hop») по маршруту от источника к приемнику.

Если бы состав опрашиваемых узлов не изменялся в ходе функционирования системы, то параметры пакетов MAINTENANCE можно было бы рассчитать заранее и включить их в программу статически. Такой вариант не подходит для программ с самоконтролем, поскольку они разрабатываются в расчете на непредсказуемые изменения условий функционирования. Использование виртуального образа RapidIO позволяет формировать пакеты MAINTENANCE динамически с учетом того, какое множество коммутаторов нужно опросить в текущий момент. Виртуальный образ отображает текущее состояние сети RapidIO. Узлы, признанные «сбойными», помечаются в виртуальном образе специальным признаком и исключаются в последующих циклах опроса.

Будем далее называть «виртуальным сканированием» RapidIO опрос узлов виртуального образа RapidIO, сопро-

вождаемый формированием пакетов MAINTENANCE для отправки в физические узлы. В зависимости от решаемой задачи в процессе виртуального сканирования может проводиться отбор узлов для физического опроса по тем или иным критериям (по типу узла или по другим метаданным).

4.2 Организация сканирования сети. Процессоры-сканеры и сканируемые множества.

Далее будем называть процессоры, с которых производится сканирование сети RapidIO, «процессорами-сканерами».

В общем случае возможна ситуация, когда при установленных маршрутах не все коммутаторы доступны с единого процессора. Мы применяем следующий подход. Сеть RapidIO сканируется с нескольких процессоров-сканеров. В качестве сканеров используются все процессоры, участвующие в решении целевой задачи. Для каждого сканера определяется множество сканируемых коммутаторов RapidIO (далее - «сканируемое подмножество»), для каждого узла формируется набор пакетов MAINTENANCE для опроса его регистров. Сканируемые подмножества не пересекаются. Поставленная задача решается динамически на каждом процессоре с использованием виртуального сканирования. Ниже приведено описание использованного нами алгоритма.

Инициализация процедуры:

Перебор процессоров по логическому номеру i. Для процессора с логическим номером i создаем сканируемое множество и включаем в него единственный элемент: сам процессор. Приписываем соответствующему виртуальному узлу метку «i».

Внешний цикл:

Инкрементация числа hop на единицу, начиная с нуля.

Внутренний цикл:

Перебор процессоров по логическому номеру i.

Шаг i:

Выбираем i-й процессор и виртуальным сканированием определяем все доступные ему узлы с дальностью hop («фронт hop i-го узла»). Иначе говоря, делаем hop шагов по каждому маршруту, исходящему из i-го узла. Приписываем метку «i» всем ранее не помеченным узлам на фронте hop и включаем эти узлы в i-е сканируемое множество.

Завершение процедуры:

Процедуру продолжать до тех пор, пока на шаге hop не выполнится одно из условий:

1) все узлы RapidIO помечены,
2) для каждого процессора фронт hop пуст.

Следует иметь в виду следующую проблему. В общем случае может оказаться, что какой-либо коммутатор доступен только с непроцессорного оконечного устройства (например, с микросхемы 1890ВГ18Я, используемой в режиме ВСК [5]).

Решение проблемы в общем случае пока не найдено. Исследуем ее в частном случае: для архитектуры модулей, разрабатываемых в ФГУ ФНЦ НИИСИ РАН. Единственное непроцессорное оконечное устройство RapidIO – это микросхема 1890ВГ18Я. Она входит в состав модулей ЦПРИО-128, МР-РИО и мезонинного модуля ВСК2 БТМ24-302. Коммутаторы с потенциально возможным отсутствием доступа: 1890КП3Я (в составе модулей ЦП128-РИО, ЦП64-РИО, МР-РИО) и коммутатор процессора 1890ВМ8Я (в составе модулей ЦП21, ЦП22). Коммутатор на процессорном модуле доступен с процессорных элементов этого модуля, т.к. только через него возможен выход на RapidIO. Остается коммутатор 1890КП3Я на модуле МР-РИО. Пример, когда возможно отсутствие доступа в коммутатор 1890КП3Я, приведен на рис.3.

4.3 Диагностика перехода соединения SRIO из режима 4x в режим 1x на маршруте потока данных в цикле конвейера

Рассмотрим одну из возможных причин замедления передачи данных по mp-каналу.

Маршрут передачи данных от источника к приемнику проходит через последовательность портов коммутаторов и гибридных устройств. В этой последовательности могут встречаться порты двух типов: последовательные (SRIO [1]) и параллельные (PRIO [1]). Под «соединением» понимается пара соединяемых портов смежных устройств. Порты SRIO есть у микросхем 1890ВГ18Я (коммутатор SRIO <-> PRIO) и 1890ВМ6Я (универсальный процессор КОМДИВ 64-РИО, далее - К64). Порт SRIO может находиться в одном из двух режимов: 1x и 4x. Штатно используется режим 4x. При технических сбоях соединение может самопроизвольно переключаться в режим 1x. Такое переключение приводит к снижению скорости

передачи данных. Пропускная способность SRIO в режиме 1x составляет 125 Мбайт/с в одном направлении. Падение скорости передачи данных по mp-каналу ниже этой величины служит основанием проверить:

- встречаются ли соединения SRIO на маршруте от источника к приемнику,
- если такие соединения есть, то не переключилось ли одно из них в режим 1x.

Рис.3 Пример коммутатора, через который не проходит маршрут с процессора

Режим соединения SRIO определяется текущим режимом соединяемых микросхем. Текущий режим микросхемы 1890ВГ18Я или К64 можно определить по состоянию ее регистров.

Мы предлагаем следующий подход:

-опрашивать только те порты SRIO, через которые проходят потоки данных прикладной задачи;

- в состав параметров самоконтроля ввести «скорость передачи данных по mp-каналу» для всех выходных портов потоков данных БПОС [2];
- скорость измерять на каждой итерации конвейерного цикла и сравнивать с пороговым значением (110 Мбайт/сек);
- при падении скорости ниже порогового значения последовательно опрашивать состояние портов SRIO на маршруте mp-канала до первого обнаружения порта с режимом 1x.

Примечание. Проверка описанного подхода проводилась с использованием экспериментальной версии операционной системы, в которой была добавлена возможность измерять время передачи сообщения MESSAGE [1] от момента отправки запроса на передачу данных до момента подтверждения завершения транзакции. Эта возможность использовалась в программе данных самоконтроля для измерения скорости передачи данных по mp-каналу.

Программная реализация приведенного подхода потребовала доработки экспери-ментальной версии БПОС и программы сбора данных самоконтроля [2]. В БПОС была расширена структура порта потока данных путем включения в нее дополнительных полей для измеренной скорости передачи данных и виртуального идентификатора сбойного коммутатора RapidIO. В программе сбора данных самоконтроля была доработана функция statistic_send() [2]. Это функция вызывается потоком БПОС по завершении итерации конвейерного цикла на процессоре. Функция последовательно опрашивает порты потоков БПОС. В доработанной версии для каждого выходного порта функция вычисляет скорость передачи данных, сравнивает его с порогом и принимает решение о необходимости сканирования маршрута. При положительном решении выполняется виртуальное сканирование маршрута:

- из структуры порта извлекается логический номер процессора-приемника,
- по виртуальному образу определяется РИО-идентификатор приемника,
- в виртуальном образе перебираются коммутаторы по маршруту от источника к приемнику с последовательно возрастающим числом hop,
- для каждого коммутатора определяется его тип (PRIO/SRIO),
- если тип коммутатор SRIO, то:
- в физический коммутатор отправляется пакет MAINTENANCE для выяснения установленного режима (1x/4x). Параметры пакета MAINTENANCE (РИО-идентификатор приемника и число hop) задаются в соответствии с их текущими значениями,

- если порт коммутатора находится в режиме 1x, то в структуру порта потока БПОС записывается виртуальный идентификатор коммутатора,

- процедура продолжается до первого обнаружения порта SRIO в режиме 1x или до исчерпания маршрута.

Виртуальные идентификаторы коммутаторов SRIO, пребывающих в режиме 1x, передаются на аккумулирующий процессор функцией statistic_send() в составе данных самоконтроля.

Отметим, что можно было бы решить задачу без виртуального сканирования. Можно последовательно опрашивать пакетами MAINTENANCE все физические узлы на маршруте. РИО-идентификатор процессора-приемника можно узнать по его логическому номеру и по массиву RIO_MAP[7], число hop последовательно инкрементировать. Но способ с виртуальным сканированием лучше. Его преимущество - сокращение количества пакетов MAINTENANCE, циркулирующих в физической среде RapidIO. Пакеты отправляются только в те физические узлы, где есть смысл искать проблему.

4.4 Сканирование сети RapidIO на произвольной частоте

В текущей версии программы сбора данных самоконтроля сканирование RapidIO применяется для решения следующих задач:

- опрос счетчиков переданных данных в портах коммутаторов,

- выявление отсутствующих (нарушенных) физических соединений,

- обнаружение перехода коммутаторов SRIO в режим 1x.

Ниже перечислены принципы организации сканирования.

Сканирование должны осуществлять все процессоры, выполняющие прикладную программу.

В образ ОСРВ процессора-сканера должны быть включены: файл bmrw, библиотека БПОС, библиотека сбора данных самоконтроля (СОСД).

На процессоре должен быть запущен отдельный поток-сканер с пониженным приоритетом.

В пользовательской функции main() средствами СОСД должен быть создан виртуальный образ RapidIO.

При инициализации потока-сканера программа СОСД методом виртуального сканирования определяет сканируемое подмножество коммутаторов RapidIO для процессора-сканера и создает набор пакетов MAINTENANCE для опроса этих коммутаторов.

Поток-сканер на заданной частоте производит опрос своего сканируемого подмножества коммутаторов. Частота сканирования может отличаться от частоты вычислительного конвейера.

Для опроса узлов RapidIO в них направляются пакеты MAINTENANCE. Ответы сохраняются в виртуальном образе RapidIO. В виртуальном образе каждому узлу соответствует своя структура с полем «массив портов». У виртуального порта два поля. В них помещаются ответы от физического узла на пакеты MAINTENANCE. Первое поле – значение счетчика переданных через порт данных. Второе поле – признак действительности соединения. По завершении опроса портов физического узла поток-сканер выставляет в структуре виртуального узла признак «данные обновлены».

В программе СОСД введен новый тип данных [2]: «результаты сканирования». Этот тип описывает результаты опроса порта узла RapidIO. Протокол передачи данных на аккумулирующий процессор доработан для передачи результатов сканирования.

По завершении итерации задачи БПОС поток задачи вызывает функцию СОСД statistic_send(). Первоначальное назначение функции – отправка системных и пользовательских данных самоконтроля [2] на аккумулирующий процессор. Функция доработана. Теперь она наряду с системными и пользовательскими данными собирает и отправляет обновленные результаты сканирования. По завершении вычитывания портов виртуального узла функция statistic_send() сбрасывает его признак «данные обновлены».

На аккумулирующем процессоре стадия-аккумулятор обрабатывает результаты сканирования посредством пользовательского обработчика соответствующего типа данных [2].

5 Заключение

Реализованы два варианта опроса коммутаторов в сети RapidIO 1) сканирование маршрута передачи данных между двумя процессорами в случае снижения скорости передачи ниже порогового уровня, 2) сканирование сети RapidIO на заданной частоте.

Сканирование сети RapidIO осуществляют все процессоры, выполняющие прикладную программу. Каждый процессор-сканер опра-

шивает свое множество коммутаторов. В общем случае при разбиении множества коммутаторов на сканируемые подмножества могут остаться коммутаторы, к которым не проложены маршруты ни с одного процессора. В текущей версии программы такие коммутаторы исключаются из опроса.

При реализации опроса коммутаторов используется виртуальный образ сети RapidIO, отражающий ее текущее состояние. Это позволяет сократить число пакетов MAINTENANCE, циркулирующих в физической сети, а также исключить сбойные коммутаторы из числа опрашиваемых. Программы с самоконтролем рассчитаны на непредсказуемые изменения условий функционирования. В этом случае использование виртуального образа RapidIO особенно полезно, поскольку позволяет формировать пакеты MAINTENANCE динамически с учетом того, какое множество коммутаторов нужно опросить в текущий момент.

RapidIO fabric scanning in self-controlled parallel digital signal processing programs for multiprocessor real-time systems

T.K. Gringauz, A.N. Onin

Abstract: Multiprocessor real-time systems with RapidIO [1] communications are considered. Local program timing results are collected by the single accumulating processor during computing conveyor cycle [2]. Periodic RapidIO fabric scanning is added to technology [2] in order to increase self-diagnostic efficiency .RapidIO scanning method is described.

Keywords: system self-diagnostics, parallel digital signal processing library, computing pipeline stage, processor group, data flow, timer, flow port, accumulator stage, handler function, RapidIO communication environment, NWRITE transaction.

Литература

1. RapidIO Interconnect Specification (Revision 1.3) Available from: http://www.rapidio.org/specs/current.
2. Т.К. Грингауз, А.Н. Онин. Технология сбора данных самоконтроля для программ параллельной обработки сигналов в мультипроцессорных комплексах реального времени. «Труды НИИСИ РАН», т.4 (2018), №2, 155-166.
3. Райко Г.О. Библиотека параллельной обработки сигналов. «Труды НИИСИ РАН», т.5 (2015), №1, 64-69.
4. А.Н. Годунов, В.А. Солдатов. Операционные системы семейства Багет (сходства, отличия и перспективы). «Программирование», т.40 (2014), № 5, 68 – 76.
5. Т.К. Грингауз, А.Н. Онин. Программное обеспечение для приема и передачи данных по высокоскоростному каналу в мультипроцессорных комплексах реального времени. «Труды НИИСИ РАН», т.4(2014), №2, 22-32.
6. Т.К. Грингауз, А.Н. Онин. Инструментальное программное обеспечение для подготовки запуска задач в мультипроцессорных комплексах реального времени. «Труды НИИСИ РАН», т. 5 (2015), №2, 122-129.
7. Т.К. Грингауз, А.Н. Онин. Особенности использования конфигурационных файлов при интеграции технологий параллельной обработки сигналов, приема данных по высокоскоростному каналу, подготовки запуска задач в мультипроцессорных комплексах реального времени. «Труды НИИСИ РАН», т.7 (2017), №1, 58-69.
8 Т.К. Грингауз, А.Н. Онин. Технология разработки целевых программ для мультипроцессорных комплексов реального времени с приемом данных по высокоскоростному каналу. «Суперкомпьютерные технологии (СКТ-2018): V Всероссийская научно-техническая конференция. - Дивноморское, Геленджик 17-22 сентября 2018 г. (материалы конференции)», Ростов-на-Дону; Таганрог, Издательство Южного федерального университета, 2018, том 1, 76-79.

ЭФФЕКТИВНЫЙ ПОДХОД К ОРГАНИЗАЦИИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ПРИ МОДЕЛИРОВАНИИ МНОГОФАЗНОЙ МНОГОКОМПОНЕНТНОЙ ФИЛЬТРАЦИИ

И.В. Афанаскин[1], В.А. Бахтин[2], С.Г. Вольпин[3], А.В. Королев[4], Н.В. Поддерюгина[5]

[1, 3, 4] ФГУ ФНЦ НИИСИ РАН, Москва, Россия
[2, 5] ИПМ РАН, Москва, Россия

E-mail's: [1] ivan@afanaskin.ru, [2] bakhtin@keldysh.ru, [3] sergvolpin@gmail.com, [4] alexandre.korolev@mail.ru, [4] konov@keldysh.ru

Аннотация: Для трехмерной модели многофазной многокомпонентной фильтрации рассмотрен поход к созданию параллельной версии программы, которая может эффективно выполняться на кластерах с ускорителями различной архитектуры. Производительность предлагаемых конфигураций оценивается на примере моделирования разработки нефтяной залежи с помощью закачки в пласт сухого/жирного газа.

Ключевые слова: модель многокомпонентной фильтрации, автоматизация разработки параллельной версии программы, вытеснение нефти газом.

Введение

При проектировании разработки месторождений углеводородов чаще всего используются модели с упрощенным представлением пластовых флюидов, с их описанием в виде так называемых несимметричных бинарных систем. Для нефтяных месторождений пластовый газ представляется одним компонентом, а пластовая нефть - состоящей из товарной нефти и растворенного в ней газа. Для газовых и газоконденсатных месторождений пластовая жидкость состоит из товарного конденсата (получаемого в результате сепарации продукции), а пластовый газ состоит из товарного газа и растворенного в нем конденсата. Иногда при моделировании разработки газоконденсатных месторождений и месторождений с легкими летучими нефтями предпочтительнее рассматривать симметричные двухкомпонентные системы, в которых и жидкая, и газовая фаза представляются состоящими из обоих компонентов [1].

Подобные модели вполне пригодны для описания разработки месторождений углеводородов в режиме истощения, в режимах с поддержанием пластового давления за счет закачки в пласт различных агентов (воды, газа). Однако для описания более сложных процессов, сопровождающих разработку месторождений при использовании специальных методов, повышающих степень извлечения углеводородов из недр (закачка пара, термогазовое воздействие, водогазовое воздействие на пласт с достижением смесимости вытесняющего агента с пластовой нефтью, организация сайклинг-процесса и др.) такое упрощенное представление пластовой углеводородной системы лишь двумя компонентами оказывается недостаточным. В подобных случаях пластовые флюиды описываются с помощью многокомпонентной системы, в которой в качестве отдельных компонентов могут выступать как индивидуальные компоненты, так и интегрированные компоненты, например, отдельные фракции, состоящие из множества индивидуальных компонентов [2, 3].

Равновесные составы фаз, участвующих в совместной фильтрации при разработке залежи углеводородов, определяются с применением коэффициентов распределения, зависящих от термобарических условий в пласте и состава флюидов. Мощным инструментом, позволяющим выполнять расчеты равновесных состояний вплоть до критических значений давления и температуры, является применение аппарата уравнений состояния для многокомпонентной системы [3]. Использование уравнений состояния позволяет

вместе с этим получать внутренне согласованные равновесные составы и плотности фаз.

Для изучения равновесной многокомпонентной фильтрации используется математическая модель, описывающая законы сохранения количества вещества по компонентам и количества движения по фазам, представленная в виде системы дифференциальных уравнений в частных производных. Эта система замыкаются с помощью ограничений, представленных алгебраическими уравнениями, и дополняется начальными и граничными условиями. Система решается численным способом. Дифференциальные уравнения аппроксимируются методом конечных разностей. С учетом потенциально большого количества сеточных блоков и значительного числа компонентов, необходимых для представления реальных пластовых систем, применяется последовательный метод решения IMPECS (неявное давление, явные концентрации компонентов и насыщенности) [3].

Для реализации описанного подхода создана численная трехмерная модель и написана программа на алгоритмическом языке FORTRAN [6]. Для существенного ускорения расчетов процессов многокомпонентной фильтрации в случае использования значительного числа сеточных блоков с применением системы автоматизированного распараллеливания программ САПФОР [4, 5] создана многопроцессорная версия рассматриваемого симулятора. Полученную распараллеленную программу можно эффективно рассчитывать на вычислительных машинах с различной архитектурой: как на базе многоядерных универсальных процессоров, так и с использованием графических ускорителей, а также сопроцессоров. Причем программа работает без дополнительных изменений.

1. Многокомпонентная многофазная изотермическая фильтрация

Систему уравнений, описывающих изотермическую фильтрацию, представим в виде системы уравнений материального баланса с ограничениями на ряд переменных [3, 6]:

$$\frac{\partial}{\partial t}\left[\phi\left(\xi_o S_o x_i + \xi_g S_g y_i\right)\right] + \\ + div\left(\xi_o x_i \vec{W}_o + \xi_g y_i \vec{W}_g\right) - q_i = 0, \\ i = 1,2,\ldots, N_c, \quad (1)$$

$$\frac{\partial}{\partial t}\left[\phi \xi_w S_w\right] + div\left(\xi_w \vec{W}_w\right) - q_w = 0, \quad (2)$$

$$\vec{W}_\alpha = -k k_{r\alpha} \mu_\alpha^{-1}\left(grad P_\alpha + \rho_\alpha g\, grad D\right), \\ \alpha = o, g, w, \quad (3)$$

$$f_i^o - f_i^g = 0,\ i = 1,2,\ldots, N_c, \quad (4)$$

$$\sum_{i=1}^{N_c} x_i = 1,\ \sum_{i=1}^{N_c} y_i = 1, \quad (5)$$

$$S_o + S_g + S_w = 1, \quad (6)$$

$$P_o - P_w = P_{cwo},\ P_g - P_o = P_{cog}, \\ P_g - P_w = P_{cwo} + P_{cog}, \quad (7)$$

здесь N_c – суммарное число компонентов в углеводородных (нефтяной и газовой) фазах, k – абсолютная проницаемость пласта, ϕ – открытая пористость пласта, t – время, P_α – давление для фазы α, P_{cwo} и P_{cog} – капиллярное давление в системах вода-нефть и нефть-газ, ξ_α и ρ_α – молярная и массовая плотность фазы α, μ_α – динамическая вязкость фазы α, $k_{r\alpha}$ – относительная фазовая проницаемость, q_i и q_w - плотность источников и стоков для компонентов и воды, $D(x,y,z)$ – разница глубин между точкой пласта с координатами (x,y,z) и заданной плоскостью, g - ускорение свободного падения, f_i^o и f_i^g - летучести компонента i в нефтяной и газовой фазах.

Исходная система (1)-(3) с учетом соотношений (7) преобразуется к виду:

$$\frac{\partial}{\partial t}\left[\phi\left(\xi_o S_o x_i + \xi_g S_g y_i\right)\right] - \\ div\left[\xi_o x_i k k_{ro} \mu_o^{-1}\left(grad\, P + \rho_o g\, grad D\right) + \right. \\ + \xi_g y_i k k_{rg} \mu_g^{-1}\left(grad\, P - grad\, P_{cgo} + \right. \\ \left. \left. + \rho_g g\ \ grad D\right)\right] - q_i = 0, \quad i = 1,2,\ldots, N_c, \quad (8)$$

$$\frac{\partial}{\partial t}\left[\phi \xi_w S_w\right] - div\left[\xi_w k k_{rw} \mu_w^{-1}\left(grad\, P - \right.\right. \\ \left.\left. - grad\, P_{cwo} + \rho_w g\, grad D\right)\right] - q_w = 0. \quad (9)$$

Уравнения баланса количества компонентов для углеводородных фаз (8) можно переписать, как:

$$\frac{\partial}{\partial t}\left[\phi\, z_i\left(\xi_o S_o + \xi_g S_g\right)\right] -$$
$$- div\left[\xi_o x_i k k_{ro} \mu_o^{-1}\left(grad P + \rho_o g\, grad D\right) +\right.$$
$$+ \xi_g y_i k k_{rg} \mu_g^{-1}\left(grad(P - P_{cgo}) +\right.$$
$$\left.\left. + \rho_g g\, grad D\right)\right] - q_i = 0,$$

(10)

$$z_i = x_i L + y_i V,\ i = 1,2,...,N_c,$$

$$L = \frac{\xi_o S_o}{\xi_o S_o + \xi_g S_g},\ V = \frac{\xi_g S_g}{\xi_o S_o + \xi_g S_g},$$

$$\sum_{i=1}^{N_c} z_i = 1, \qquad (11)$$

где P – давление в нефтяной фазе, L и V – мольные доли жидкой и газовой фаз, а z_i - мольная концентрация i-го компонента в углеводородной системе.

Такая запись удобна при использовании разностной схемы неявной по давлению и явной по концентрациям и насыщенности (IMPECS). Складывая по i уравнения (10) и добавляя к ним уравнение для водной фазы (9) получают уравнение для давления. Иногда уравнение (9) берут с весом θ:

$$\frac{\partial}{\partial t}\left[\phi\left(\xi_o S_o + \xi_g S_g + \theta \xi_w S_w\right)\right] -$$
$$- div\left[\xi_o k k_{ro} \mu_o^{-1}\left(grad P + \rho_o g\ grad D\right) +\right.$$
$$+ \xi_g k k_{rg} \mu_g^{-1}\left(grad(P - P_{cgo}) + \rho_g g\ grad D\right) +$$
$$\left. + \theta \xi_w k k_{rw} \mu_w^{-1}\left(grad(P - P_{cwo}) + \rho_w g\ grad D\right)\right] -$$
$$- q_o - q_g - \theta q_w = 0.$$

(12)

Давление на новом шаге по времени определятся неявным образом. В результате формируются нелинейные уравнения, которые линеаризуются методом Ньютона. При этом производными по мольным концентрациям пренебрегается.

Мольные плотности компонентов, а затем и их мольные доли z_i определяем с помощью уравнения (10) применив к нему явную схему. Используя методы расчета равновесия фаз при известных давлении, температуре и мольных долях компонентов углеводородную смесь разделяем на газовую и нефтяную фазы определяя x_i, y_i, L и V.

Затем последовательно определяем насыщенности по фазам S_w, S_o и S_q с помощью уравнений (2) и (11).

Использование уравнений состояния для многокомпонентной смеси даёт возможность получить внутренне согласованные равновесные составы и плотности фаз. Среди известных уравнений состояния в нефтегазовой отрасли при расчетах широко применяются следующие: Соаве-Редлиха-Квонга, Пенга-Робинсона, Редлиха-Квонга.

Система уравнений для изотермической композиционной модели совместной фильтрации нефти, газа и воды, решается с учетом заданных начальных и граничных условий.

2. Примеры расчетов вытеснения нефти газом

В НИИСИ РАН для реализации описанного подхода к решению задач изотермической фильтрации многокомпонентных многофазных систем флюидов была создана численная трехмерная модель и написана программа на алгоритмическом языке FORTRAN (далее программа «NCOM») [6]. Общее количество строк программы составляет около 14 000, общее количество циклов в программе более 1000.

Для отработки методики автоматизированного распараллеливания программы создана её последовательная версия на языке Fortran-DVMH [4]. Чтобы оценить затраты времени, необходимого для выполнения различных фрагментов программы, выполнили расчеты двух вариантов разработки залежи нефти с помощью системы газонагнетательных и добывающих скважин (5-точечная система с плотностью 50 га/скв), в пласт закачивается сухой и жирный газ.

В начальном состоянии в пласте присутствует «связанная» вода (неподвижная вода, не участвующая в фильтрации) и нефть, которую можно описать в виде девятикомпонентной системы с индивидуальными компонентами: метан CH4, этан C2H6 и т.д. и псевдокомпонентами для описания более тяжелых фракций нефти (C15, C20).

Составы сухого газа, жирного газа и нефти представлены в таблице 1. Закачка газов разного состава сопровождается в значительной степени отличающимися процессами вытеснения нефти газом.

Таблица 1

Составы нефти (О) и газов (сухого G1 и жирного G2)

М. доля	C1	C2	C3	C4	C5	C6	C10	C15	C20
O	0,4000	0,0800	0,0700	0,0450	0,0250	0,0750	0,1050	0,1000	0,1000
G1	0,9992	0,0001	0,0001	0,0001	0,0001	0,0001	0,0001	0,0001	0,0001
G2	0,6500	0,1250	0,1100	0,0650	0,0345	0,0140	0,0005	0,0005	0,0005

При закачке сухого газа в окрестности нагнетательных скважин формируется двухфазная область совместного течения нефти и газа, которая расширяется с увеличением объема закачки. Газ в пласте при этом значительно отличается по составу: вблизи нагнетательных скважин он близок к составу закачиваемого газа, а вблизи фронта вытеснения в значительной степени обогащен промежуточными фракциями.

При закачке жирного газа в нефть конденсируются промежуточные компоненты, создавая возможность ее смесимости с газом, который закачивается в нагнетательные скважины. То есть, возможно формирование составов, соответствующих однофазному, закритическому состоянию флюида. В окрестности этой области газовая и жидкая фазы сближаются по составу и физическим свойствам (плотность, вязкость). При этом снижается величина поверхностного натяжения между двумя фазами, что меняет характер вытеснения нефти газом (изменяется вид кривых относительных фазовых проницаемостей и капиллярного давления со снижением величины остаточной нефтенасыщенности).

Различный характер вытеснения нефти газами с разным составом удобно проиллюстрировать с помощью треугольных диаграмм, на которых девятикомпонентные системы представлены в виде трехкомпонентных с объединением легких (C1), промежуточных (C2–C6) и тяжелых (C7+) компонентов. Для пластовой температуры T=98°C на диаграммах представлены области с двухфазным поведением пластовой системы при трех существенно различных значениях пластового давления: начального (P=244 атм, рис.1) и двух значений, близких к давлению нагнетания газа (P=400 атм): P=375 атм, рис. 2 и P=385 атм, рис. 3. На диаграммах также представлены составы сухого (G1) и жирного (G2) газов, состав исходной пластовой нефти (O) и критические точки (C). Линии, которые разделяют области двухфазного и однофазного состояния флюидов, соответствуют точкам кипения и точкам росы при заданных значениях температуры и давления; они сходятся в критической точке C. Отрезки прямых линий, соединяющие равновесные составы жидкой и газовой фазы (коноды), также нанесены на диаграммах. В критических точках проведены касательные прямые, соответствующие предельным положениям конод с их приближением к критической точке.

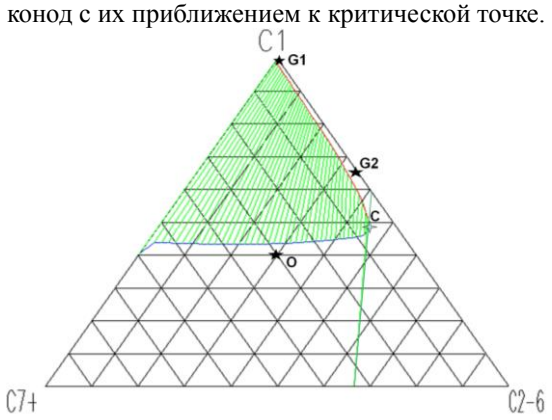


Рис. 1. Фазовая диаграмма при T=98°C, P=244 атм

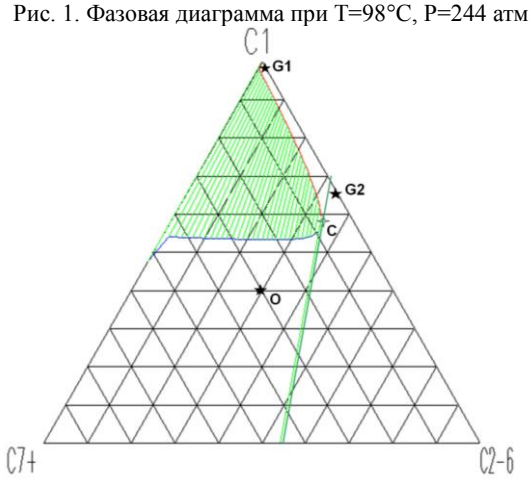


Рис. 2. Фазовая диаграмма при T=98°C, P=350 атм

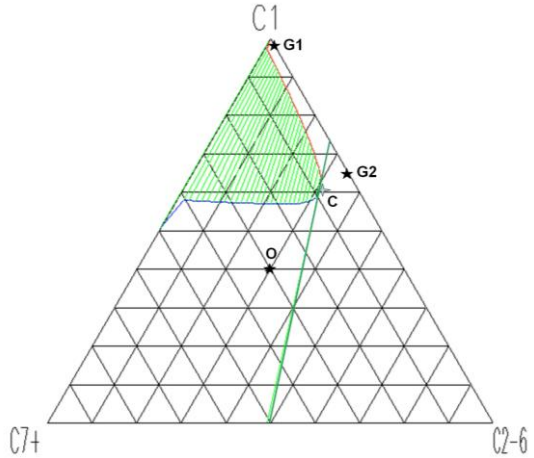


Рис. 3. Фазовая диаграмма при T=98°C, P=385 атм

Для вариантов расчета с различными составами нагнетаемого газа характерно различное положение точек G1, G2 и O по отношению к касательной, отделяющей докритическую область от закритической. При закачке сухого газа точки G1 и O лежат в докритической области по одну сторону от касательной для всего диапазона пластовых давлений. При закачке жирного газа точки G2 и O при высоких давлениях лежат по разные стороны от касательной и для некоторых смесей нагнетаемого газа и пластовой нефти жидкая фаза за фронтом вытеснения может смешиваться с набегающим газом, оставаясь в однофазном (закритическом) состоянии.

В плане вычислений отличие составов нагнетаемого газа приводит к различиям в логической схеме: закачка сухого газа приводит к образованию двухфазной области и в расчетах фазовых равновесий составы фаз на старом временном слое служат хорошим начальным приближением для процедуры определения составов на новом временном слое; при закачке жирного газа вероятно попадание составов в закритическую область. Требуется выделять данную ситуацию, сохраняя при этом «хорошее» приближение, чтобы использовать его в дальнейшем при возможном возвращении в докритическую область (при соответствующем изменении состава смеси и давления).

Свойства пласта: эффективная мощность – 10 м, проницаемость пласта – 10,4 мД, коэффициент вертикальной анизотропии проницаемости – 0,1 д.ед., открытая пористость – 0,09 д.ед. Расстояние между рядами нагнетательных и добывающих скважин при квадратной сетке бурения – 500 м, шаг вычислительной сетки по осям X и Y совпадает и составляет – 100 м, шаги по оси Z – 1 м, 4 слоя по 2 м и 1 м. Начальное давление в пласте 244 атм. В дальнейшем забойное давление на добывающих скважинах – также 244 атм. Забойное давление в нагнетательных скважинах – 400 атм. Водонасыщенность на начальный момент времени 0,1 д.ед., что соответствует насыщенности связанной водой.

На рис. 4 представлен расчетный элемент симметрии для пятиточечной системы разработки, на рис. 5-6 представлены распределение насыщенности газом и распределение зон докритического (0) и закритического (1) состояния пластовой углеводородной системы через 6 лет при закачке в пласт жирного газа.

Динамика объемов годовой и накопленной закачки сухого газа (в расчете на одну нагнетательную скважину), а также годовой и накопленной добычи нефти и газа (в расчете на одну добывающую скважину) представлена на рис.7-8.

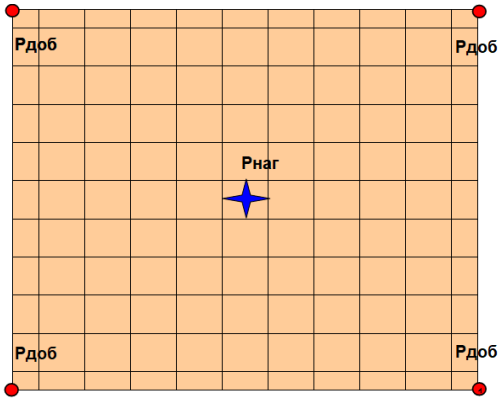


Рис. 4. Элемент симметрии 5-точечной системы разработки

11	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	75	0	0	0	0	0
8	0	0	0	90	78	90	78	90	0	0	0
7	0	0	0	78	90	90	90	78	0	0	0
6	0	0	75	90	90	90	90	90	75	0	0
5	0	0	0	78	90	90	90	78	0	0	0
4	0	0	0	90	78	90	78	90	0	0	0
3	0	0	0	0	0	75	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
X--->	1	2	3	4	5	6	7	8	9	10	11

Рис. 5. Распределение газонасыщенности (%) на t=6 лет

11	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	1	1	0	1	1	0	0	0
8	0	0	1	1	0	0	0	1	1	0	0
7	0	0	1	0	0	0	0	0	1	0	0
6	0	1	0	0	0	0	0	0	1	0	
5	0	0	1	0	0	0	0	0	1	0	0
4	0	0	1	1	0	0	0	1	1	0	0
3	0	0	0	1	1	0	1	1	0	0	0
2	0	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
X--->	1	2	3	4	5	6	7	8	9	10	11

Рис. 6. Распределение зон докритического (0) и закритического (1) состояния углеводородной системы на t=6 лет

Для сравнения на рис. 9-10 приводится динамика объемов годовой и накопленной закачки жирного газа (в расчете на одну нагнетательную скважину), а также годовой и накопленной добычи нефти и газа (в расчете на одну добывающую скважину).

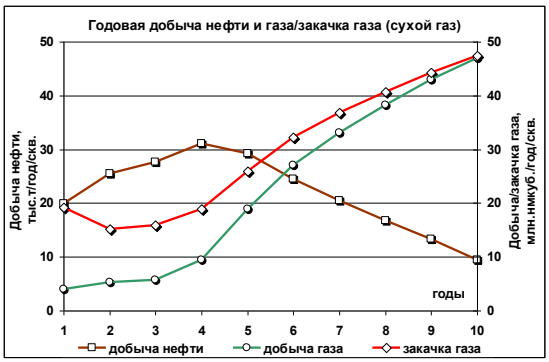


Рис. 7. Вариант с закачкой сухого газа.
Годовая закачка газа, годовой отбор нефти и газа

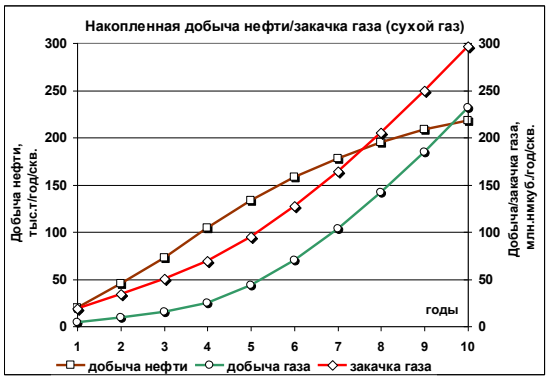


Рис. 8. Вариант с закачкой сухого газа. Накопленная закачка газа, накопленный отбор нефти и газа

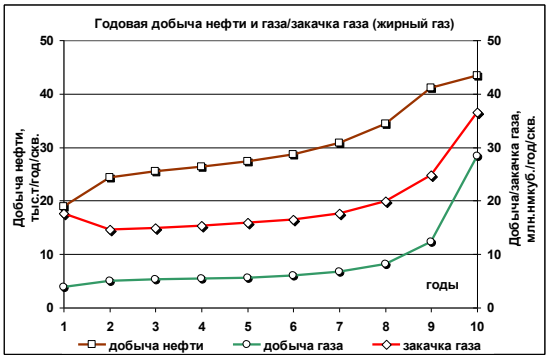


Рис. 9. Вариант с закачкой жирного газа.
Годовая закачка газа, годовой отбор нефти и газа

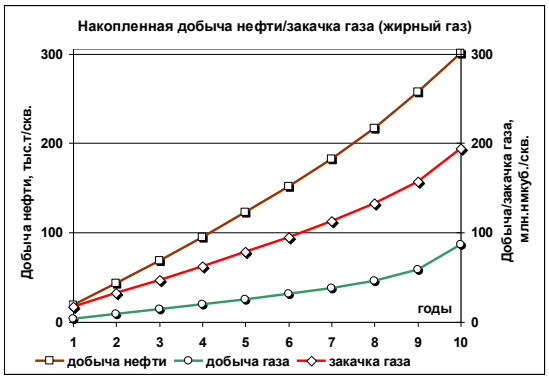


Рис. 10. Вариант с закачкой жирного газа.
Накопленная закачка газа, накопленный отбор нефти и газа

3. Разработка параллельной версии программы

Наибольшая сложность разработки параллельной версии программы для кластера – необходимость принимать глобальные решения, связанные с распределением данных, а также вычислений при учете свойств всей программы. Потом необходимо выполнить кропотливую работу по изменению программы и по ее отладке. Принятие решений по согласованному распределению данных и вычислений затрудняется большим объемом программного кода, многомодульностью и многофункциональностью.

Для решения данной проблемы в системе САПФОР [4] реализован метод инкрементального распараллеливания. Его главная идея состоит в том, что распараллеливанию подвергается не вся программа, а ее отдельные части - в них заводятся дополнительные экземпляры требуемых данных, выполняется распределение этих данных и соответствующих вычислений. Для взаимодействия с не подвергавшимися распараллеливанию частями программы применяются операции копирования исходных (нераспределенных) данных в дополнительные (распределенные) данные и обратно.

Для определения времяемких фрагментов программы NCOM, требующих распараллеливания, использовалось профилирование. Чтобы более точно оценить затраты времени на выполнение отдельных блоков программы в процессе расчета варианта разработки с вытеснением нефти газом при использовании 5-точечной системы скважин в расчетной модели элемент симметрии пятиточечной системы повторен многократно по направлениям X и Y. Таким образом, общий размер модели составил 261x261x6=408 726 сеточных блоков, число добывающих скважин – 729, нагнетательных – 676. Скважины начинают работать одновременно с условиями управления, соответствующими их статусу. В этих условиях решения имеют симметрию, что позволяет оценивать правильность и расчетов их точность при выборе метода решения системы линейных алгебраических уравнений, а также при задании параметров, влияющих на корректность вычисления равновесий фаз.

На рис. 11 показано в секундах время выполнения процедур программы для расчета варианта с закачкой жирного газа. Эти величины получены на суперкомпьютере К-100 (ИПМ) при использовании анализатора производительности, входящего в состав DVM-

системы. Если не учитывать процедуры, в которых выполняется ввод/вывод (REMAPI и REMAPR), то основное время занимают процедуры: COMPOZ, SXYN, SXYDEF, LSOR. На первом этапе были распараллелены именно эти процедуры.

На рис. 12 показано время выполнения (в секундах) на суперкомпьютере К-100 частично распараллеленной программы при использовании различного числа вычислительных узлов для расчета варианта с закачкой жирного газа.

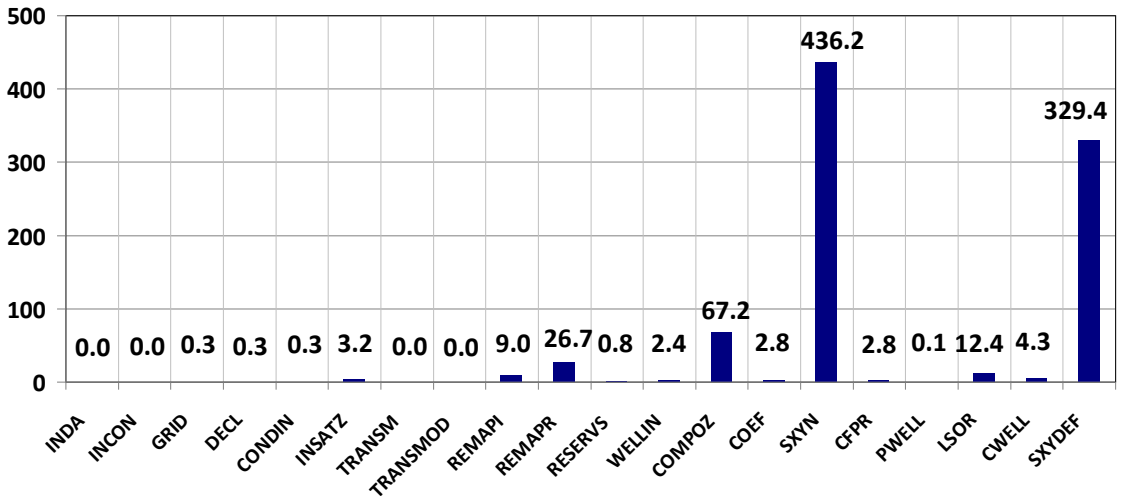


Рис. 11. Время выполнения различных процедур программы, секунды

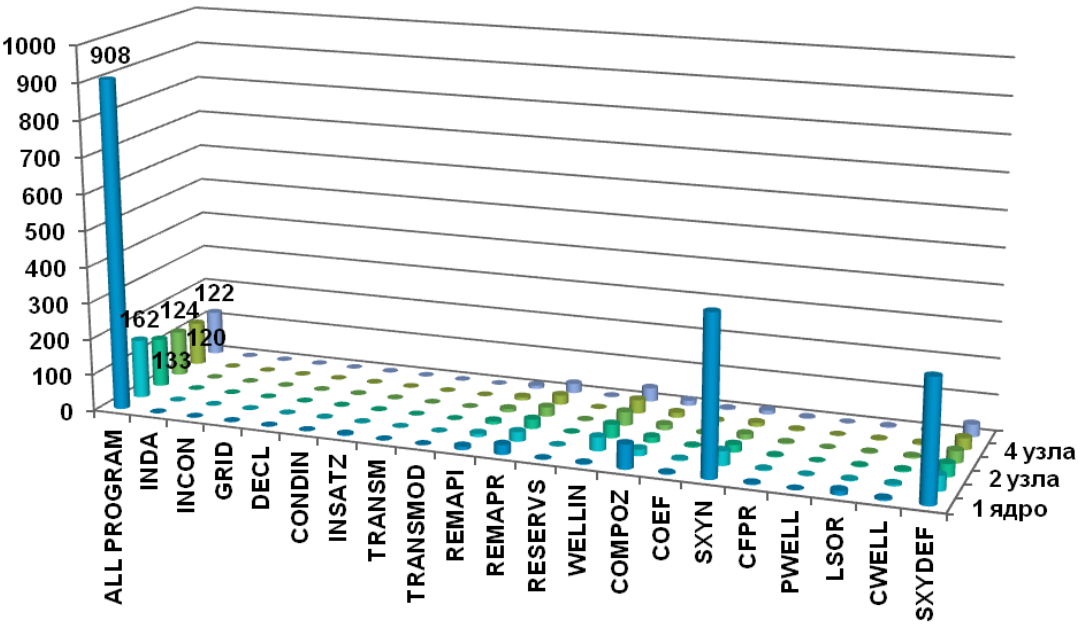


Рис. 12. Время выполнения частично распараллеленной программы на К-100, секунды

При использовании четырёх вычислительных узлов К-100 расчёт ускоряется в 7,5 раз по сравнению с расчётом на одном ядре (время счета сокращается с 908 до 120 секунд). При дальнейшем увеличении числа используемых узлов расчёт замедляется. Основная причина – рост накладных расходов на копирование данных из распределенных массивов в исходные нераспределенные массивы, выполняемое после завершения выполнения параллельной версии процедуры. Лишь полное распараллеливание программы позволяет избавиться от операций копирования.

Если распараллеливание основных процедур (2700 из 13844 строк) программы фактически не требовало изменения текста

последовательной программы и было выполнено достаточно быстро, то полное распараллеливание потребовало серьезного изменения структуры программы: инлайн подстановки процедур (для фрагментов программы, выполняемых на ускорителях), преобразования операторов ввода/вывода (DVMH-модель накладывает ограничения на использование распределенных массивов в операторах ввода/вывода), перехода на динамические массивы (вместо их моделирования) и других. Найденные при распараллеливании основных процедур решения по распределению данных были использованы при распараллеливании других частей программы.

Полное распараллеливание программы позволяет запускать ее в различных режимах. Режим работы DVMH-программы, количество используемых нитей, графических процессоров задается при помощи переменных окружения и не требует перекомпиляции программы. На рис. 13-14 показано время выполнения (в секундах) 100 шагов по времени программы в режиме MPI/OpenMP на суперкомпьютерах MVS-10P (МСЦ), K-100 (ИПМ) и Ломоносов (МГУ) при использовании от 1 до 8 вычислительных узлов для вариантов расчета с закачкой сухого и жирного газа.

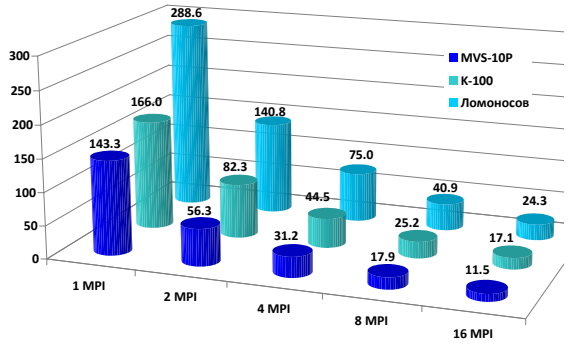


Рис. 13. Время выполнения 100 шагов параллельной программы на разном числе узлов для варианта расчета с закачкой сухого газа, секунды

На каждом узле запускался 1 или 2 MPI-процесса, каждый из которых создавал 8 (для MVS-10P), 6 (для K-100) или 4 OpenMP-нити. Всего для расчетов использовалось до 128 ядер ЦПУ. При увеличении числа используемых ядер в 16 раз расчет ускоряется в 10-13 раз.

На рис 15-18 время расчета 100 временных шагов при закачке сухого и жирного газа сопоставляется для сеток с возрастающим числом блоков в модели: 261x261x6, 521x521x6, 1041x1041x6 на суперкомпьютерах K-100 и Ломоносов.

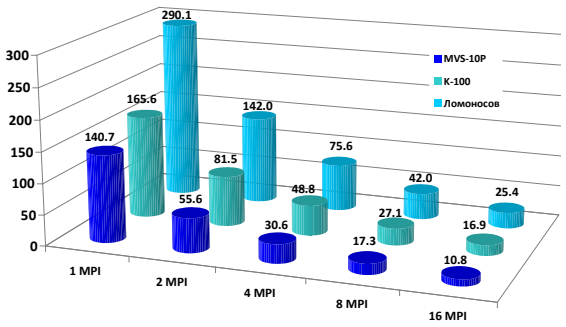


Рис. 14. Время выполнения 100 шагов параллельной программы на разном числе узлов для варианта расчета с закачкой жирного газа, секунды

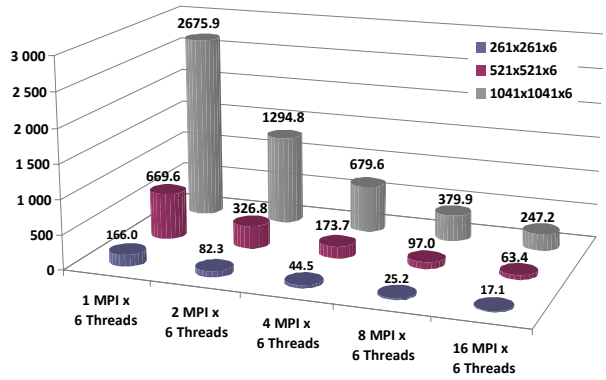


Рис. 15. Время выполнения 100 шагов параллельной программы для разного числа сеточных блоков на разном числе узлов для варианта расчета с закачкой сухого газа, секунды. Суперкомпьютер K-100 (ИПМ)

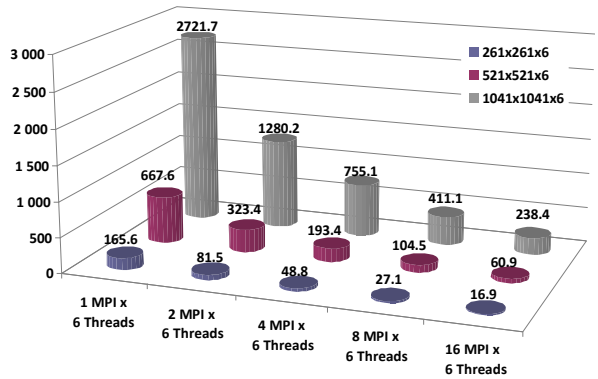


Рис. 16. Время выполнения 100 шагов параллельной программы для разного числа сеточных блоков на разном числе узлов для варианта расчета с закачкой жирного газа, секунды. Суперкомпьютер K-100 (ИПМ)

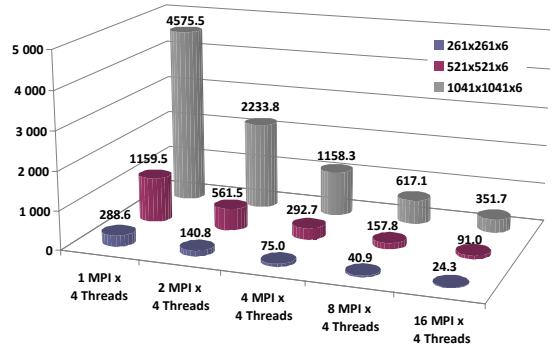


Рис. 17. Время выполнения 100 шагов параллельной программы для разного числа сеточных блоков на разном числе узлов для варианта расчета с закачкой сухого газа, секунды. Суперкомпьютер Ломоносов (МГУ)

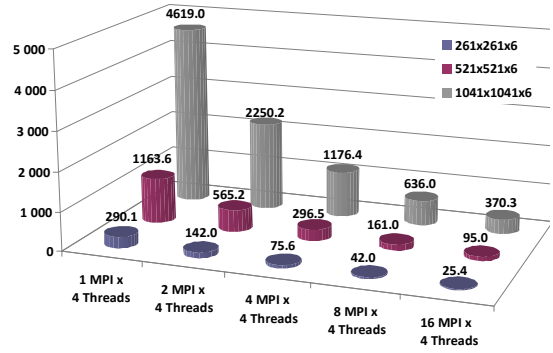


Рис. 18. Время выполнения 100 шагов параллельной программы для разного числа сеточных блоков на разном числе узлов для варианта расчета с закачкой жирного газа, секунды. Суперкомпьютер Ломоносов (МГУ)

На рис. 19-20 показано время выполнения полного расчета программы на исходной сетке на суперкомпьютерах K-100 и K-10 (ИПМ) в случае использования графических ускорителей. Синим цветом отмечено время, полученное при использовании только ядер ЦПУ, желтым – графических ускорителей, зеленым – при использовании ядер ЦПУ совместно с ускорителями [5].

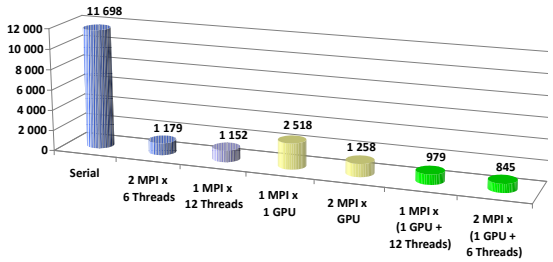


Рис. 19. Время выполнения программы в различных режимах на суперкомпьютере K-100, секунды

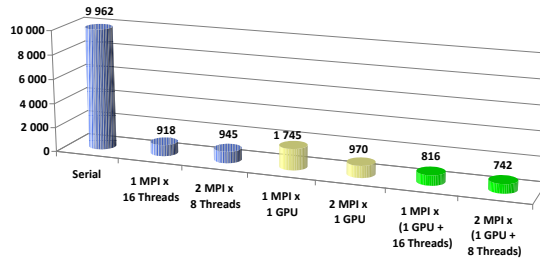


Рис. 20. Время выполнения программы в различных режимах на суперкомпьютере K-10, секунды

Использование одного графического ускорителя сокращает время выполнения программы в 4,6-5,7 раз по сравнению с расчетом на одном ядре. Однако при этом расчёт выполняется медленнее чем, при использовании всех ядер ЦПУ узла. Основная причина в том, что не удается сбалансировать нагрузку нитей (вычисления в различных блоках сетки, вблизи которых находятся добывающие и нагнетательные скважины, существенно отличаются). Получить ускорение при использовании графических ускорителей удается за счет совмещения вычислений на ЦПУ и ГПУ. В таком режиме расчёт ускоряется с 11698 секунд до 845 секунд на K-100; с 9962 секунд до 742 секунд на K-10. Таким образом, при использовании всех вычислительных устройств одного узла ускорение расчёта составляет 13,84 раза для K-100 и 13,42 раза для K-10 по сравнению с расчётом на одном ядре.

Заключение

Для численной трехмерной модели и программы для расчета многофазной многокомпонентной фильтрации разработана параллельная версия, которая может эффективно выполняться на кластерах разной архитектуры, использующих универсальные многоядерные процессоры, графические ускорители, а также их комбинации. При этом отображенные на узел вычисления могут с учетом производительности автоматически распределяться между вычислительными устройствами узла.

Эффективность различных конфигураций оценивается на примере моделирования разработки нефтяной залежи с применением закачки в пласт сухого/жирного газа на сетках большого размера.

Работа выполнена при поддержке гранта РФФИ 16–29–15105 офи_м.

An Efficient Approach to the Organization of Parallel Computations for Multiphase Multicomponent Flow Simulation

I.V. Afanaskin, V.A. Bahtin, S.G. Volpin, A.V. Korolev, N.V. Podderyugina

Abstract: Development of parallel version of program for three-dimensional multiphase multicomponent flow model that can be efficiently performed on clusters with accelerators of different architecture is described. Two cases of oil field development with dry/wet gas injection were used to estimate the efficiency of various configurations.

Keywords: multicomponent flow simulator, automatic development of parallel program version, oil displacement by gas.

Литература

1. Х. Азиз, Э. Сеттари. Математическое моделирование пластовых систем. М., Недра, 1982.
2. H. Kazemi, C.R. Vestal and G.D. Shank. An Efficient Multicomponent Numerical Simulator. «Soc. Pet. Eng. J.», (1978), Oct., 355-368.
3. L.X. Ngiem, D.K. Fond and K. Aziz. Compositional Modeling with an Equation of State. «Soc. Pet. Eng. J.», (1981), Dec., 687-698.
4. В.А. Бахтин, М.С. Клинов, В.А. Крюков и др. Расширение DVM-модели параллельного программирования для кластеров с гетерогенными узлами. «Вестник Южно-Уральского университета», серия «Математическое моделирование и программирование», т. 277 (2012), № 18, 82-92.
5. В.А. Бахтин, М.С. Клинов, А.С. Колганов и др. Автоматическое отображение Фортран-программ на кластеры с ускорителями. «Научный сервис в сети Интернет: Международная суперкомпьютерная конференция – Новосибирск, 22-27 сентября 2014 г. (труды)», М., Изд-во МГУ, 2014, 17-22.
6. И.В. Афанаскин, С.Г. Вольпин, А.В. Королев. Использование метода последовательных приближений при композиционном моделировании разработки месторождений углеводородов для определения концентраций компонентов. «Труды НИИСИ РАН», т. 8 (2018), № 2, 33-43.

Отбор данных в системе складского учета и способы подборов компонент

Г.Л. Левченкова

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

Аннотация: Рассматриваются способы поиска данных и принципы выбора компонент с использованием информации, введенной в систему складского хранения/перемещения материальных ценностей.

Ключевые слова: складская система, хранение, перемещение, поиск, отбор, учет

1. Введение

Настоящая статья является продолжением описания работы в складской системе, представленной в публикации [1].

В работе [1] были рассмотрены принципы занесения информации в базу данных и особенности хранения материальных ценностей. Данная статья посвящена организации процесса поиска и выдачи требующихся товаров. Возможности складской системы обеспечивают проведение предварительного анализа при подборе компонент, а также оформление списков для работников склада, занимающихся компоновкой наборов материальных ценностей, подлежащих выдаче. В статье описываются методы отбора сведений и действия персонала, связанные с оформлением передач со склада.

2. Запрос

Для того, чтобы материальные ценности (МЦ) могли быть выданы, необходим соответствующий документ. Документ может быть устным распоряжением, или служебным документом, или каким-либо указанием другого типа, но в любом случае очевидно, что работники склада по своей инициативе ничего выдавать не имеют права.

Назовем любое требование на выдачу со склада запросом. В запросе должна содержаться обязательная информация и могут присутствовать уточняющие данные. Вне зависимости от полноты поступивших данных, складские работники должны найти и выдать нужный товар и занести соответствующие сведения в складскую систему. Если запись, соответствующая выданной МЦ, окажется не полностью заполненной обязательными к заполнению данными, то в системе предусмотрен "признак незавершенности записи". Взведя этот признак, можно будет вернуться к окончательному оформлению строки позже, не задерживая процесс выдачи.

Особым случаем запроса является расформирование (разукомплектование) изделия. Такое действие так же, как и в случае подбора, влечет за собой изменения строк базы данных. Процесс поиска товара и оформления записей аналогичен запросу на выдачу, но имеет особенности, которые будут рассмотрены далее.

Полученный запрос инициирует поиск и выдачу/переоформление товара. Складской работник прежде всего анализирует наличие следующей ***необходимой информации***:

1) Поступивший запрос должен быть зарегистрирован в подразделении (в данном случае на складе), как входящий. Другими словами, пришедшему запросу (будь то служебная записка или запись просьбы, поступившей по телефону) должен быть присвоен входящий номер. Это элементарное требование, но многие его игнорируют. Именно этот номер будет занесен в складскую систему в качестве основания для выдачи в группу полей подбор/зарезервировано: название потребности.

2) Наименование и количество МЦ.

Здесь могут присутствовать уточнения:

- возможные аналоги;
- нужно ли некоторое количество "лишних" компонент в упаковке?;

- необходимо ли некоторое количество пустых позиций в упаковке? Комплектующие могут быть мелкими, и для удобства их установки/пайки на автоматах иногда требуется, чтобы лента/паллета, в которую упакованы компоненты, была длиннее, чем количество занятых компонентами мест.

Помимо перечисленных выше двух обязательных пунктов в запросе может присутствовать ***дополнительная информация***:

1) Как срочно следует выдать МЦ.

2) На какой участок выдается подбор или ФИО сотрудника, которому следует выдать товар.

3) Ожидать ли перемещения "по бухгалтерии", или товары впоследствии вернутся на склад?

4) Допустима ли замена МЦ аналогичным товаром из другой партии?

3. Подбор

После того как запрос получен, принят и зарегистрирован, следует приступить к анализу и выбору нужных МЦ. Порядок действий следующий: сначала отбирается необходимая информация из базы данных, потом производится выбор подходящих для выдачи записей о товаре, затем записи корректируются (делятся) таким образом, чтобы можно было распечатать только нужные строки для осуществления подбора. Отобранные строки впоследствии будут переведены в "архивное состояние" (в складской системе), а работник склада сможет по списку выбрать компоненты.

3.1 Поиск информации и анализ строк базы данных

Запросы на выдачу могут иметь разную степень сложности.

В самом простом случае однозначно известны все перечисленные ниже данные, уже имеющиеся в складской системе:

"назначение" (определяемое при оформлении товара на склад);

- наименование (без аналогов) и количество (без дополнительных компонент).

Более сложный вариант, когда к требуемому количеству добавлено требование выдать либо ленту нужного размера, либо "лишние" компоненты.

К еще более сложному подбору относится случай возможной выдачи аналогов товара.

наименование по складу	код товара по складу	количество (к учету)	признак совместного хранения	упаковка			количество запасных	партия		признак незавершенности записи	подбор/зарезервировано	назначение	название потребности	место расположения/хранения
				признак упаковывания вместе с МЦ из другой партии	количество мест в ленте/паллете	кол-во: всего в упаковке		количество в партии	кодовое слово идентификатор партии					
Микросхема Ф	46299	25	V	U_105_18	лента 65	60		25	237_N18	230519 p_056_19	p_056_19	Изделие Ч	сборка	шкаф 33, ящик 67
Микросхема ФА	46299	2			лента 10	2		2	112_N16		p_056_19	Изделие Ч	резерв	шкаф 33, ящик 67
Микросхема Ф	46299	100	V		лента 150	120	20	100	105_N15	230519 p_056_19	p_056_19	Изделие Ч	сборка	шкаф 33, ящик 67
Микросхема Ф	46299	35	V	U_105_18	лента 65	60		35	200_N18	230519 p_056_19		Изделие Х		шкаф 33, ящик 67
Микросхема ФИ	46299	90			лента 90	90		100	111_N19		p_056_19	Изделие Ч	сборка	шкаф 33, ящик 67
Микросхема ФИ	46299	10			лента 20	10		100	111_N19			Изделие У		шкаф 33, ящик 67
Микросхема ФИ	46299	30			лента 40	30		35	211_N19			Изделие У		шкаф 33, ящик 67
Микросхема ФИ	46299	5	V	U_122_19	лента 15	15		35	211_N19	230519 p_056_19	p_056_19	Изделие Ч	сборка	шкаф 33, ящик 67
Микросхема ФИ	46299	15			лента 20	15		25	213_N19			Изделие Е		шкаф 33, ящик 67
Микросхема ФИ	46299	10	V	U_122_19	лента 15	15		25	213_N19	230519 p_056_19		Изделие Е		шкаф 33, ящик 67

Рис.1. Отбор записей с нужным наименованием

ВАЖНО: У всех строк, которые будут подвергаться любым изменениям, связанным с составлением списка для подбора, следует установить признак незавершенности записи.

Первый случай. Простой вариант подбора. Известны наименование, количество и, возможно, "назначение".

Производится отбор строк по наименованию либо "коду товара по складу". Если указано значение поля "назначение", то ставится фильтр на "назначение". В результате получается список из строк с нужным наименованием, и проставив в соответствующее поле признак "подбора" (в группу полей подбор/зарезервировано), можно будет распечатать листок для работника склада, который достанет соответствующие упаковки с полок (см. Рис. 1).

Пусть в подбор заказано выдать товара с кодом 46299 в количестве 222 штуки: 220 для последующей сборки изделий + 2 резервных.

Для работника склада будет распечатан следующий список (см. Рис. 2).

Второй случай. Более сложный вариант подбора. Требуется выдать ленту нужной длины, "с запасом" либо "лишние компоненты".

Требование выдать ленту нужной длины означает, что подобрать нужно одним "куском", а не набирать заданное количество несколькими партиями.

Лента "с запасом" означает, что количество мест в ленте должно превышать количество товара в этой ленте. То есть число посадочных мест должно быть больше, чем количество упакованного в ленту товара.

Просьба выдать "лишние" компоненты означает, что какое-то количество компонент в упаковке должны быть из числа "запасных". Например, для подбора в 250 штук (в том числе потребность составляет 217 шт. + приблизительно 33 запасных) будет сначала анализироваться наличие 217-ти учтенных и минимум 33-х запасных. Затем будет искаться лента нужной длины (например, не менее 260 посадочных мест), в которой будет не менее 250-ти штук товара.

Отобрав строки с нужным товаром, работник склада анализирует группу полей Упаковка и в ней колонку "кол-во всего в упаковке". Наверняка найдется не одна запись, в которой требуемое к выдаче количество (с запасом) покрывается наличием целого "куска" (лентой/паллетой), от которого можно отделить запрашиваемое количество компонент. Для выбора записи, от которой будет отделяться требуемое количество, следует руководствоваться следующими принципами:

- лучше выбрать для подбора строку, в которой содержится требуемое количество и нет признака хранения с МЦ из другой партии, либо отделяемое количество позволит этот признак снять. Тогда потом будет проще осуществить переоформление строк;
- при разделении ленты следует учитывать количество остающихся на складе компонент: отрезать лучше от большего куска. Если же товар штучный, и не требуется выдать "ленту нужной длины", то предпочтительнее выдать несколько мелких партий целиком, чем

наименование по складу	код товара по складу	количество (к учету)	место расположения/хранения	Примечание	упаковка			признак совместного хранения	количество запасных	партия		признак незавершенности записи	подбор/зарезервировано	назначение	название потребности
					кол-во: всего в упаковке	признак упаковывания вместе с МЦ из другой партии	количество мест в ленте/паллете			количество в партии	кодовое слово идентификатор партии				
Микросхема Ф	46299	25	шкаф 33, ящик 67	отделить ровно 25 шт. в ленте 25	60	U_105_18	лента 65	V		25	237_N18	230519 p_056_19	p_056_19	Изделие Ч	сборка
Микросхема ФА	46299	2	шкаф 33, ящик 67		2		лента 10			2	112_N16		p_056_19	Изделие Ч	резерв
Микросхема Ф	46299	100	шкаф 33, ящик 67	отделить ровно 100 шт. в ленте 100	120		лента 150	V	20	100	105_N15	230519 p_056_19	p_056_19	Изделие Ч	сборка
Микросхема ФИ	46299	90	шкаф 33, ящик 67		90		лента 90			100	111_N19		p_056_19	Изделие Ч	сборка
Микросхема ФИ	46299	5	шкаф 33, ящик 67	отделить ровно 5 шт. в ленте 5	15	U_122_19	лента 15	V		35	211_N19	230519 p_056_19	p_056_19	Изделие Ч	сборка

Рис.2. Простой подбор

наименование по складу	код товара по складу	количество (к учету)	место расположения/хранения	Примечание	упаковка			признак совместного хранения	количество запасных	партия		признак незаверш-енности записи	подбор/зарезер-вировано	назначение	название потребности
					кол-во: всего в упаковке	признак упаковывания вместе с МЦ из другой партии	количество мест в ленте/паллете			количество в партии	кодовое слово идентификатор партии				
Конденсатор CC	77081	217	шкаф 62, ящик 99	отделить ровно 250 шт. в ленте 260	300		лента 380	V	50	250	116_N15	230519 p_056_19	p_056_19	Изделие Ч	сборка
Конденсатор EE	77022	200	шкаф 61, ящик 12	отделить ровно 250 шт. в ленте 260	750	U_112_18	лента 780	V	50	600	288_N18	230519 p_056_19	p_056_19	Изделие Ч	сборка

Рис.3. Подбор с "запасом"

отделять запрашиваемое количество от большой партии;

- если количество запрошенной МЦ ненамного меньше общего количества, упакованного в ленту, то следует выдать ленту целиком, а не отрезать от неё кусок. Впоследствии этот излишек будет возвращен на склад, и возвращенная лента будет иметь достаточную длину благодаря наличию пустых мест в упаковке, несмотря на малое количество товара, который в ней остался.

Для работника склада будет распечатан следующий список (см. Рис. 3).

Третий случай. Вариант подбора, в котором допустимы замены компонент на аналоги.

В некоторых запросах наряду с требованием выдать большой кусок ленты могут быть уточнения, что в случае отсутствия данного наименования допустимо выдать аналог компоненты. Главное, чтобы лента оказалась нужной длины, либо надо обязательно набрать определенное количество штук. В этих случаях для просмотра полной картины наличия придется задействовать несколько механизмов отбора в несколько этапов.

Первый этап: установить особый признак подбора у всех подходящих компонент. Механизм: установка фильтра, запись особого признака подбора в подходящие записи, снятие фильтра. Фильтры на записи, в частности, могут устанавливаться по "части наименования", по коду товара по складу, по "назначению", по количеству в упаковке, и т.д.

Второй этап состоит в отборе записей по проставленному особому признаку, их анализе, выборе нужных строк путем проставления признака текущего подбора и в снятии "особого признака" у всех строк, которые не удовлетворяют требованиям подбора.

В случае, если выбран аналог вместо

наименование по складу	код товара по складу	количество (к учету)	место расположения/хранения	Примечание	упаковка			признак совместного хранения	количество запасных	партия		признак незаверш-енности записи	подбор/зарезер-вировано	назначение	название потребности
					кол-во: всего в упаковке	признак упаковывания вместе с МЦ из другой партии	количество мест в ленте/паллете			количество в партии	кодовое слово идентификатор партии				
Соединитель DS	12824	25	шкаф 56, ящик 11	отделить ровно 25 шт. (вместо Соединителя НН)	26	U_005_18	лента 26	V		25	007_N18	230519 p_056_19	p_056_19	Изделие Ч	сборка
Соединитель TT	12800	175	шкаф 78, ящик 67	(вместо Соединителя НН)	175		лента 175			175	077_N18	230519 p_056_19	p_056_19	Изделие Ч	сборка
Соединитель НН	12500	200	шкаф 19, ящик 19		200		лента 200			200	177_N19	230519 p_056_19	p_056_19	Изделие Ч	сборка

Рис.4. Подбор с аналогами

запрашиваемого товара, в поле Примечание заносится информация о "наименовании в запросе на выдачу", чтобы можно было понять, какая компонента заменена данной упаковкой.

Для работника склада будет распечатан следующий список (см. Рис. 4).

3.2 Подбор компонент

После всех манипуляций с фильтрацией записей, анализом информации по каждой компоненте, выбором нужных количеств и изменением содержания полей в зависимости от потребностей подбора, в определенном месте каждой записи появится признак текущего подбора. Этот признак дает возможность распечатать в виде таблицы сводную информацию для работника склада, который будет заниматься комплектацией подбора.

Список для подбора содержит следующую информацию, которая необходима работнику для того, чтобы вынуть товар с полки:

- место расположения;
- наименование по складу;
- количество в упаковке;
- количество мест в ленте/паллете;
- количество запасных (в упаковке);
- примечание: сколько штук нужно отделить из данной упаковки.

После того, как требуемое количество компонент отделено от упаковки, работник склада, осуществляющий подбор, должен сделать следующее:

- на упаковке, которая возвращается на склад, нужно, при необходимости, изменить надпись: проставить новое количество, кол-во в упаковке и кол-во запасных компонент (если оно было);
- на упаковке, которая отделена в подбор, указать кол-во штук, если оно изменилось;
- в списке "для подбора" сделать соответствующую отметку "подобрано" и указать, при необходимости, "новое" количество оставшихся штук или мест в ленте, если оно изменились при пересчете (чтобы потом занести эту информацию в базу данных);
- указать в списке новое расположение остающегося на складе товара, если оно подлежит изменению. Например, большая партия компонент раньше хранилась в коробке на стеллаже, а теперь оставшиеся 2 штуки вполне уместятся в ящик на полке в шкафу;
- в ящик с подбором вложить список подобранного.

После того, как нужные изделия будут лежать в ящике для выдачи, база данных может потребовать корректировки в части записей, относящихся к данному подбору.

3.3 Изменение строк базы данных

Для отбора строк базы данных, требующих корректировки, следует воспользоваться возможностью выбора записей по ранее установленному признаку незавершенности записи у всех строк, которые изменялись.

По завершении работы по разделению/отделению нужного количества компонент работником склада, осуществляющим подбор, нужно откорректировать отобранные записи согласно комментариям, сделанным в "списке подбора". После пересчета может измениться количество, или товары, ранее хранящиеся в одной коробке, теперь окажутся отдельно упакованными. Может потребоваться перемещение остатка компоненты в другое место хранения с оформлением нового местоположения товара в базе данных.

Это самая кропотливая и требующая аккуратности часть работы по учету МЦ - занесение информации в записи о товаре, который нуждается в разбиении упаковки на части.

Первоначальные манипуляции с записями и товаром уже сделаны работником склада, осуществляющим подбор:

1) найден товар, записи о котором подлежат редактированию, в базе данных и на полке (в руки взята упаковка);
2) взятая упаковка содержит именно столько МЦ, сколько получается при сложении количеств в выбранных строках базы данных;
3) от упаковки отделено (или сделана пометка на упаковке в случае резервирования) то количество товара, которое следует выдать/зарезервировать.

Далее начинается работа за компьютером:

1) записи разбиваются (строки копируются и редактируются "количества по складу") таким образом, чтобы разбиение МЦ на новые упаковки (или назначения) соответствовало этим записям;

2) в случае выдачи запасных компонент одним куском с учтенными, в записи которых отсутствовали "лишние" компоненты, придется повозиться с корректировкой полей тех записей, в которых были эти "лишние", чтобы количества учтенных товаров не изменились. Напомним также, что в момент выдачи с "запасом неучтенных компонент" следует проверить корректность заполнения группы полей "упаковка" у записей, которые связаны признаком "совместное хранение" и отражают общее количество "лишних" МЦ в упаковке. В запись, предназначенную к выдаче, заносится фактическое количество передаваемых (отрезаемых от мотка) запасных компонент, то есть выдается 200 учтенных, но отрезается кусок ленты с 230 штуками, 30 из которых лежат как "лишние". А в записи, где учтен оставшийся от партии товар, заносится "новое количество лишних".

3) у каждой записи/упаковки фиксируется изменившееся "количество дыр в ленте/ячеек в паллете";

4) если в отобранных строках был установлен "признак упаковывания вместе с МЦ из другой партии (накладной)", и к выдаче предназначен весь "кусок из другой партии" целиком, и теперь "остаток" будет размещаться на полке отдельно, то данный признак нужно снять у всех строк. После разделения в каждой из упаковок остались товары, которые получены по одному приходному документу;

5) если после обработки записей было произведено разделение упаковок, то у каждой "теперь отдельно упакованной партии" следует снять "признак совместного хранения". Напомним, что этот признак устанавливается при совместном упаковывании МЦ, записи о которых имеют какие-то отличительные черты, например, назначение, гарантия, названия документов, даты, примечания и т.п.;

Заметим, что остальные поля (за исключением, конечно, содержащих серийные номера изделий, новое распределение и новое "месторасположение") у записей, которые были затронуты при операциях подбора и резервирования, остаются без изменений.

Перед окончательным изменением информации в складской системе необходимо убедиться в том, что оставшиеся на полке компоненты помечены соответственно скорректированным записям. А именно, что признаки совместного упаковывания, признаки партии, количество лишних компонент в ленте и, возможно, примечания занесены корректно на все упаковки, которые подвергались переупаковыванию.

Все вышеописанное относится к записям о компонентах, которые остаются на хранении. Поэтому после занесения в эти строки актуальной информации флажок "признак незавершенности записи" можно снять.

Записи, относящиеся к извлеченным с полок товарам, имеют также и другой признак: признак подбора/резервирования (номер подбора). Признак незавершенности записи у этих строк будет снят только после того, как подбор будет выдан и все соответствующие выдаче поля записей будут заполнены.

Как было замечено ранее, особым видом запроса является разукомплектование (доукомплектование, переукомплектование) изделия. "Видоизменение" товара можно подразделить на три случая:

Первый случай. Требуется разделить товар на части, каждая из которых не является самостоятельным изделием (не подлежит учету). Например, рассмотрим товар "крепеж батарейки". Он состоит из двух частей: металлическая часть, которая должна припаиваться к печатной плате, и пластиковая накладка (крышечка), которая будет предотвращать выпадение батарейки. В подбор на участок пайки должна попасть металлическая часть, а пластиковая накладка будет лежать на складе, пока не потребуется выдача на участок монтажа. Таким образом, информация об МЦ распадается на две записи. В данном случае целесообразно установить в раздвоившихся записях признак совместного хранения, который в нашем случае будет "признаком временного разделения", и в поле Примечание указать части, на которые распался товар.

Особенностью данного "разделения" является то, что по бухгалтерии каждая из частей не имеет собственной цены. То есть в данном случае мы провели не разукомплектование, а именно разделение товара. Мало того, даже если составные части, получившиеся в результате разделения, имеют свое наименование по складу, то в данном случае на учете останется именно "крепеж батарейки", и не появятся товары "металлическая часть" из состава... и "пластиковая накладка" из состава...

Второй случай. Требуется разукомплектовать изделие, каждая составляющая которого подлежит самостоятельному учету на складе (имеет своё наименование и цену по бухгалтерии), например, на учете стоит ЭВМ в составе которой есть монитор, системный блок, клавиатура и мышь.

В таком "подборе" запись о товаре распадется на несколько строк, в каждой из которых будет указано "новое наименование", новое месторасположение, новый код по складу, и следует не забыть указать в группе полей "принято к учету" ссылку на акт (документ) разукомплектования и "старое наименование изделия", которое существовало до того, как из него были извлечены какие-то составные части.

Третий случай. Требуется доукомплектовать изделие компонентом, который числится на складе, как отдельный товар либо лежит как запасной.

Конечно, от доукомплектования запасным комплектующим не изменится наименование изделия, но в примечаниях нужно будет отразить тот факт, что товар подвергся изменению, например в системный блок установили дополнительный вентилятор (из числа запасных товаров), и, таким образом, на складе теперь будет учтено три системных блока (с кодом товара 44444) , два из которых не будут иметь Примечания, а у третьего будет отметка о том, что в нем установлено два вентилятора вместо одного.

При установке в системный блок вентилятора, который стоит на учете, изменится код товара у этого системного блока, поскольку последний уже не будет совпадать по комплектации (и цене) с теми двумя блоками, которые ранее поступили на склад. Измениться также может и наименование у этого доработанного системного блока. Учтенный же вентилятор "уйдет" со склада, в записи появится отметка о доукомплектовании им другого изделия согласно документу, например, вида "акт". В Примечаниях у вентилятора следует сделать пометку, в какой именно системный блок была вставлена материальная ценность.

Изменение записей, относящихся к подбору, заканчивается проставлением нового места расположения группы подобранных МЦ в помещении склада. Так, например, подобранный комплект может находиться в коробке, которая установлена на стеллаж до тех пор, пока не придет сотрудник и не заберет подбор. У каждой записи из этой группы должен быть проставлен номер коробки и стеллажа, где теперь располагается товар.

4. Архив (выдача)

Немаловажный момент - как оформить разбиение записи при отделении некоторого количества товара от упаковки, часть которой остается на хранении, а другая часть подлежит выдаче.

Есть два варианта.

1) Можно разбить запись на две (продублировать) и в каждую запись ввести новые данные о количествах и месторасположении компоненты.

2) Поместить запись в Архив на основании выполнения подбора (заполнив поле "фактическая передача со склада: отметка о выдаче"), а на склад занести две новых записи с указанием в качестве документов поступления (поле "фактическое принятие на склад: отметка о поступлении") идентификационных характеристик подбора. И уже в новые записи вводить дальнейшие действия и нужные сведения.

Рекомендуется поступать именно вторым способом, так будет легче проследить историю "путешествия" товара с места на место, и снижается риск ошибок при введении информации.

При выдаче отдельно упакованных "лишних" (запасных) компонент заполнению подлежат только графы "отметка о выдаче". После этого запись можно переводить в "архивное состояние", проставив "признак архивности записи".

Строка с записью об изделии, которое больше не находится на полке в помещении склада, должна быть окрашена (помечена красным цветом). В случае, если все обязательные поля заполнены, "признак незавершенности записи" снимается и взводится признак "архивности записи", а впоследствии строка переносится в "архив" (при ведении соответствующего архивного перечня).

Отметим, что записей о выдаче со склада достаточно много, поэтому для облегчения восприятия рекомендуется вести архивные перечни хотя бы по годам. Отдельные архивные списки позволяют проводить поиск в том числе по "истории", а не по всей базе

данных склада, что значительно ускоряет процесс. Естественно, поиск по всей базе, включая архивы, также не исключается.

5. Заключение

В системе складского хранения/перемещения материальных ценностей, описанной в настоящей статье и в работе [1], реализованы некоторые аспекты перемещения и складирования товара, рассмотренные в книге [2], с учетом специфики относительно небольшого склада для мелкосерийного изготовления электронных устройств.

Представленный в публикациях [1], [3], [4], [5] способ организации баз данных основан на одноуровневой организации информации в виде записей. Записи, содержащие сведения о МЦ, реализованы в виде отдельных строк, и пользователю доступна таблица, состоящая из строк-записей с требуемой последовательностью колонок.

Ниже перечислены преимущества одноуровневой организации данных для ведения системы складского хранения/перемещения материальных ценностей по сравнению с системой [6]:

1) многообразие возможностей отбора нужных сведений в базе данных обеспечивает быстрый поиск необходимых товаров как состоящих на учете (в системе) так и располагающихся на стеллажах;

2) возможность наложения фильтров на строки позволяет быстро отображать необходимые для анализа сведения;

3) перестановка колонок позволяет распечатать информацию в необходимом объеме в доступном и удобном для работников склада виде;

4) возможность копирования строк/записей позволяет избежать ошибок при вводе данных;

5) одноуровневая организация информации обеспечивает возможность визуального контроля отобранных записей;

6) доступ к информации о хранящихся на складе МЦ возможен в любой момент в полном объеме, в том числе и в динамике - в процессе подборов компонент.

Data sampling in a warehouse management system and methods of component selection

G.L. Levchenkova

Abstract. The article is devoted to the methods of data search and principles of component selection using the information entered to the system for stocking and movement of material assets.

Keywords: warehouse management system, stocking, movement, searching, selection, accounting

Литература

1. Г.Л.Левченкова. Оптимизация ведения складского учета. "Труды НИИСИ РАН", т. 7 (2017), №4, 118-128.

2. Стюарт Эмметт. Искусство управления складом. Минск, "Гревцов Паблишер", 2007.

3. Г.Л. Левченкова. Система "ЗАКУПКА И РАСПРЕДЕЛЕНИЕ". Обеспечение потребностей предприятия в комплектующих, материалах и готовых изделиях. "Организация документооборота и учета. Организация закупки и распределения", Москва, НИИСИ РАН, 2006, 76-234.

4. В.Н. Давыдов, Г.Л. Левченкова. Особенности проведения ревизии склада при смене информационной системы учета. "Технологии программирования. Учетные системы. Ортонормированные системы. Сборник статей под редакцией академика РАН В.Б.Бетелина", Москва, НИИСИ РАН, 2008, 93-114.

5. Г.Л.Левченкова, А.Г. Прилипко. Перенос информации между базами данных, существующими в средах с разными операционными системами. "Труды НИИСИ РАН", т. 1 (2011), №1, 77-85.

6. 1С:Предприятие. Версия 7.7. Конфигурация "Торговля+Склад" 9.2. Описание. М., Фирма «1С», 2002.

Методы создания программного компонента для добавления новых PBR-материалов в систему моделирования 3ds Max

А.В. Мальцев[1], **Е.М. Вожегов**[2]

ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН»,
E-mail's: [1]avmaltcev@mail.ru, [2]vozhegovem@icloud.com

Аннотация: В работе предлагается методы реализации программного компонента (плагина) для широко распространенной системы компьютерного моделирования Autodesk 3ds Max. Разработанное решение обеспечивает возможность создания в данной системе трехмерных виртуальных объектов на базе собственных PBR-материалов, физические свойства которых регулируются набором параметров и текстур.

Ключевые слова: виртуальная среда, моделирование, рендеринг, трехмерный объект, материал, плагин.

Введение

Темпы развития техники и цифровых технологий в современном мире способствуют широкому распространению трехмерного компьютерного моделирования в различные научные и прикладные сферы деятельности человека. Одной из значимых среди них является область имитационно-тренажерных комплексов и систем виртуального окружения. Тренажеры, в которых реальная окружающая среда заменяется синтезируемой на компьютере, применяются при обучении водителей автотранспортных средств, летчиков, космонавтов, операторов робототехнических устройств и т.п. При этом на эффективность и результат такой формы обучения сильно влияет визуальное сходство виртуальных моделей с их реальными прототипами. Сами виртуальные модели, как правило, являются полигональными, т.е. их поверхности формируются из совокупности полигонов-треугольников. Со всеми полигонами объекта или их отдельными группами ассоциируются некоторые материалы, определяющие цвет объекта при его отображении на экране или ином устройстве вывода. Поэтому двумя ключевыми факторами реалистичности виртуальной модели являются степень детализации геометрии (количество треугольников) и используемые материалы.

Один из актуальных подходов в процессе создания дизайнером трехмерных виртуальных моделей состоит в применении специальных материалов для последующей визуализации по технологии PBR (*physically based rendering*). Такие материалы позволяют характеризовать физические свойства поверхностей объектов, например, отражательную способность, диффузный цвет, шероховатость и др. Числовые значения данных характеристик для каждого конкретного материала (сталь, золото, дерево и т.п.) задаются с помощью набора параметров и текстур. Проблема состоит в том, что не все системы трехмерного компьютерного моделирования имеют в своем арсенале PBR-материалы или достаточное количество их типов.

В данной работе предлагаются методы и подходы для добавления новых типов материалов, задающих физические свойства поверхностей виртуальных объектов, в широко распространенную систему моделирования Autodesk 3ds Max посредством технологии плагинов.

1. Реализация плагина

Предлагаемый подход для внедрения нового типа материала в систему трехмерного моделирования 3ds Max основан на применении входящего в ее состав скриптового языка MAXScript [1, 2]. Пример программного кода, использующего данный язык, проиллюстрирован на рисунке 1.

Структура данных создаваемого материала наследуется от стандартного класса *Material* и его модуля *StandardMaterial*, который характеризуется некоторым набором свойств. Чтобы сохранить какой-либо требуемый

```
plugin material PBR
name:"PBR"
classID:#(0x3b906009, 0x61f643c6)
extends:StandardMaterial
replaceUI:true
(
local R1, R2, R3
fn getColMaterialTypes =
(
return #("Cook-Torrance")
)
parameters P1 rollout:R1
(
M_SHADER_TYPE type:#integer ui:cb_shType default:1
M_TWO_SIDED type:#boolean animatable:false ui: twoSided default: false
on M_TWO_SIDED set val do delegate.twoSided = val
)
rollout R1 "Shader Basic Parameters" category:1
(
dropDownList cb_shType  items:(getColMaterialTypes()) default:1 align:#center
   across:2
checkbox twoSided "2-Sided" offset:[50,4]
on cb_shType selected i do
(
if cb_shType.selection == 1 then (
addRollout R2
addRollout R3
)else(
removeRollout R2
removeRollout R3
)
)
)
parameters P2 rollout:R2
(
M_COLOR type:#color default: gray ui:col
on M_COLOR set val do delegate.diffuse = val
M_ROUGHNESS type:#float default:0 ui:roughness
on M_ROUGHNESS set val do delegate.UserParam0 = val * 100
M_METALNESS type:#float default:0 ui:metalness
on M_METALNESS set val do delegate.UserParam1 = val * 100
M_IOR type:#float animatable:true default:1.52 ui:ior
on M_IOR set val do delegate.ior = val * 2
M_ILLUMINATION type:#float default:0 ui:illum
on M_ILLUMINATION set val do delegate.selfIllumAmount = val * 100
M_OPACITY type:#float default:1 ui:trans
on M_OPACITY set val do delegate.opacity = val  * 100
)
rollout R2 "Cook-Torrance Basic Parameters" category:2
(
colorpicker col "Base Color:" height:16 fieldwidth:53 across:2 align:#right
spinner roughness "Roughness: " fieldwidth:45 range:[0,1,0] scale:0.01
spinner metalness "Metalness: "  fieldwidth:45  range:[0,1,0] scale:0.01  across:2
spinner ior "IOR: "  fieldwidth:45 range:[0.1,50,1.52] scale:0.01
spinner illum "Self-Illumination: "  fieldwidth:45 range:[0,1,0] scale:0.01  across:2
spinner trans "Opacity: "  fieldwidth:45 range:[0,1,1] scale:0.01
)
```

Рис. 1. Пример программного кода
на скриптовом языке MAXScript

параметр внедряемого материала, необходимо создать новую переменную. Ее тип данных должен соответствовать значению параметра (вещественное число, целое число, логический или текстура). Для связи переменной с некоторым свойством *P* модуля *StandardMaterial* используется обработчик событий *Set*. Доступ к этому свойству осуществляется посредством конструкции *delegate.P*.

Пользовательский интерфейс, который будет виден дизайнеру в окне *Material Editor* системы моделирования 3ds Max, создается на базе так называемых «свитков». Свиток представляет собой разворачивающуюся панель, содержащую специальные элементы управления – виджеты. Последние позволяют устанавливать и регулировать параметры материала, а также подключать необходимые текстуры. Пример окна со свитками (*Shader Basic Parameters*, *Cook-Torrance Basic Parameters* и *Maps*) показан на рисунке 2.

При открытии добавленного материала (в окне *Material Editor*) происходит инициализация всех связанных с ним виджетов. Их размеры и положения внутри полей свитков автоматически инициализируются значениями по умолчанию, что может приводить к неудачному расположению элементов управления и даже их пересечению. Для решения этой проблемы необходимо реализовать программную настройку каждого виджета. В ней выполняется передача элементу необходимых значений размера и положения (по горизонтали и вертикали), рассчитываемых на базе размеров родительского окна. Данные размеры можно получить с использованием скриптового API.

2. Параметры PBR-материала

При выполнении рендеринга виртуальных объектов по технологии PBR их принято подразделять на диэлектрики и металлы. Такой подход основан на существовании между ними значительных различий в природе отражения световых лучей. Металлы хорошо отражают падающий на их поверхность свет, полностью поглощая преломленный. Это выражается в отсутствии диффузной составляющей в отраженном световом потоке. Диэлектрики же,

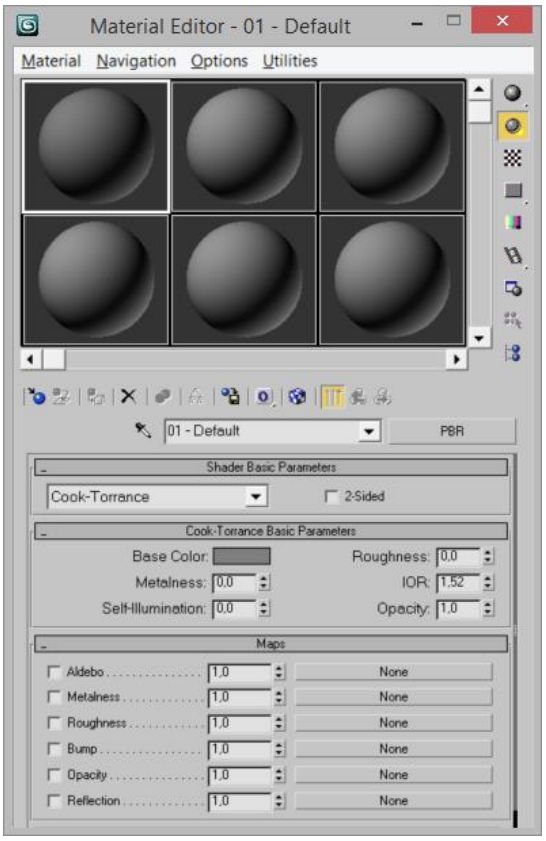


Рис. 2. Настройки добавленного PBR-материала
в системе 3ds Max

напротив, обладают низкой отражательной способностью. Преломленные лучи многократно переотражаются в материале и выходят недалеко от точек преломления, создавая диффузный свет.

Описанная классификация объектов реализуется в PBR-материалах с использованием карты металличности (*metallic*), задающей диэлектрические и металлические участки поверхности виртуального объекта. Данная текстура представляется в оттенках серого цвета (белый – чистый металл, серый – загрязненный или окислившийся металл, черный – диэлектрик). Его интенсивность влияет на долю зеркальной и диффузной составляющих света, отраженного от объекта. Другой важной текстурой, определяющей PBR-материал, является карта альбедо (*albedo*). Ее пикселы хранят коэффициенты диффузного отражения света (т.е. диффузный или базовый цвет) для диэлектрических участков поверхности объекта, а для металлических – значения $F0$ коэффициентов Френеля при нулевом угле падения световых лучей на поверхность.

Степень рассеивания света, который отражается от виртуального объекта, задается с помощью коэффициентов шероховатости, хранящихся в одноименной текстуре (*roughness*). Чем больше шероховатость поверхности объекта, тем шире и тусклее будут блики. Кроме того, при визуализации обычно моделируется микрорельеф. Он задается с помощью карты высот (*bump*) или нормалей (*normal*). Последняя содержит закодированные в цветовом формате векторы нормали, соответствующие точкам поверхности объекта.

Кроме зонального определения коэффициентов металличности, шероховатости и Френеля, а также диффузного цвета с помощью соответствующих текстур, PBR-материал должен предоставлять возможность настройки аналогичных числовых параметров для всего виртуального объекта, к которому применяется данный материал.

Заключение

В данной работе рассмотрена технология добавления в систему трехмерного компьютерного моделирования Autodesk 3ds Max новых материалов с физическими свойствами. На основе предложенных решений был создан программный модуль, подключаемый к 3ds Max как плагин. Данный модуль позволяет создавать для виртуальных объектов новые материалы с рядом настраиваемых физических свойств. Настройка производится с помощью набора числовых параметров и текстур (рис. 2).

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 18-07-00950.

Methods of creating software component to add new PBR-materials to modeling system 3ds Max

A.V. Maltsev, E.M. Vozhegov

Abstract: The paper proposes methods of implementing a software component (plugin) for widespread computer modeling system Autodesk 3ds Max. Developed solution provides an opportunity to create in this system three-dimensional virtual objects based on own PBR-materials whose physical properties specified by a set of parameters and textures.

Keywords: virtual environment, simulation, rendering, three-dimensional object, material, plugin.

Литература

1. MAXScript Introduction // URL: https://help.autodesk.com/view/3DSMAX/2017/ENU/?guid=__files_GUID_F039181A_C072_4469_A329_AE60FF7535E7_htm (*дата обращения: 30.05.2019*).
2. K.L. Murdock. Autodesk 3ds Max 2016 Complete Reference Guide. Autodesk Official Press, 2015.

Построение на GPU проекции максимальной интенсивности 3D скалярных полей цифровой модели керна

П.Ю. Тимохин[1], К.Д. Пантелей[2], Е.М. Вожегов[3], А.М. Трушин[4]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,
E-mail's: [1]tpyu@yandex.ru, [2]kpanteley@mail.ru, [3]vozhegovem@icloud.com, [4]leha_trushin@mail.ru

Аннотация: В данной работе рассматривается задача визуализации трехмерных скалярных полей, полученных в результате численных экспериментов на цифровых моделях керна – образцах породы нефтяного месторождения, оцифрованных с помощью рентгеновской компьютерной томографии. Предложены модифицированный метод и алгоритмы построения проекции максимальной интенсивности 3D скалярного поля на многоядерном графическом процессоре (GPU). Модификация направлена на предотвращение пропуска ячеек скалярного поля проецирующими лучами. На основе разработанных метода и алгоритмов был реализован модуль задачи для программного комплекса «Керносимулятор», созданного в ФГУ ФНЦ НИИСИ РАН. Была проведена апробация разработанного модуля на ряде скалярных полей, полученных в результате численного моделирования распределения давления в цифровой модели керна песчаника. Апробация подтвердила адекватность предложенного решения поставленной задаче.

Ключевые слова: визуализация, трехмерное скалярное поле, проекция максимальной интенсивности, цифровая модель керна, реальное время, GPU

Введение

Одним из стратегических направлений современного нефтегазового инжиниринга является разработка технологий повышения коэффициента извлечения нефти на основе моделирования подземной гидродинамики с учетом механических, термических, химических и комбинированных воздействий на пласт [1]. Важной частью этого подхода являются численные эксперименты на *цифровых моделях кернов* – образцах породы нефтяного месторождения, оцифрованных с помощью рентгеновской компьютерной томографии. Результатами таких экспериментов, в частности, являются *3D скалярные поля*, описывающие распределение в пространстве керна давления, температуры и других физических величин. Особый интерес представляют области скалярного поля с высокими значениями, т.к. они являются важными индикаторами корректности развития моделируемых процессов в цифровой модели керна.

Эффективным инструментом для выявления и исследования таких данных является визуализация проекции максимальной интенсивности (ПМИ) 3D скалярного поля в масштабе реального времени (с частотой смены кадров не менее 25 раз в секунду). ПМИ представляет собой изображение некоторой простой фигуры (обычно параллелепипеда или куба), ограничивающей скалярное поле, каждый пиксел которого закрашен цветом, соответствующим наибольшему значению поля вдоль луча «наблюдатель-закрашиваемый пиксел». Чтобы определить ячейку 3D скалярного поля с таким значением, используются различные подходы. В работах [2, 3] предлагаются алгоритмы вычисления индексов каждой следующей по лучу ячейки. В работе [4] выполняется проверка ячеек по лучу с некоторым фиксированным шагом. Авторы исследования [5] выполняют поиск ячейки с наибольшим значением с помощью октодерева.

В данной работе предлагаются модифицированный метод и эффективные алгоритмы решения этой задачи с распараллеливанием графических расчетов на GPU, которые обеспечивают построение проекции максимальной интенсивности 3D скалярного поля в масштабе реального времени с предотвращением случаев пропуска ячеек поля проецирующими лучами.

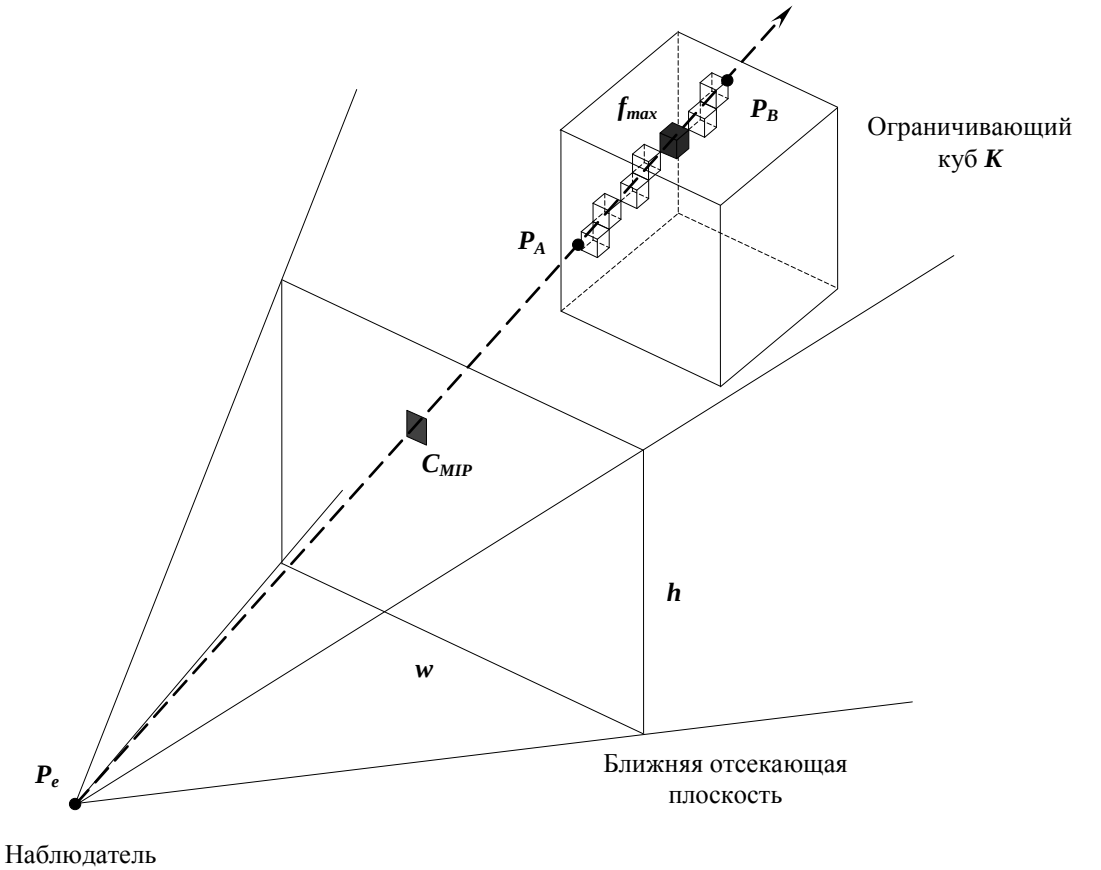


Рис. 1. Построение проекции максимальной интенсивности

1. Постановка задачи

Пусть имеется 3D скалярное поле F размером n^3 ячеек. Ограничивающим объемом данного поля в виртуальной сцене будет некоторый куб. Также в сцене имеется виртуальная камера (наблюдатель) с симметричной пирамидой видимости и областью вывода на ближней отсекающей плоскости размером $w \times h$ пикселов, где $w, h \geq n$ (см. рис. 1).

Рассмотрим некоторый пиксел области вывода, в который будет выводиться фрагмент изображения ограничивающего куба. Обозначим через P_e позицию виртуальной камеры и проведем луч $\vec{r}$ из P_e через середину рассматриваемого пиксела (далее будем называть такой луч *проецирующим*). Луч $\vec{r}$ будет пересекать ряд ячеек скалярного поля F (см. рис. 1). Обозначим через $f_{\max}$ наибольшее из значений этих ячеек, а через C_{MIP} - цвет из палитры, соответствующий значению $f_{\max}$. На рисунке 2 видно, что, если размер (длина ребра) ограничивающего куба не будет достаточно большим по отношению к области вывода, то часть ячеек не будет пересекаться ни одним проецирующим лучом, т.е. будет потеряна часть значений $f_{\max}$ скалярного поля (цветов C_{MIP}), что недопустимо. Введем пороговый размер d_{thres} куба, при котором каждую ячейку поля F, попавшую в пирамиду видимости камеры, будет пересекать хотя бы один проецирующий луч. Обозначим через K ограничивающий куб размера d_{thres}, а через I - множество фрагментов изображения куба K. Тогда, построить проекцию максимальной интенсивности скалярного поля F можно, вычислив цвет C_{MIP} для каждого фрагмента из множества I.

В данной работе решение поставленной задачи выполняется на GPU в три этапа. На ***первом этапе*** определяется функция отображения (рендеринга) куба K в множество I. На ***втором этапе*** для каждого фрагмента множества I выполняется расчет координат точки P_B выхода проецирующего луча из куба K (см. рис. 1). На ***третьем этапе*** для каждого фрагмента множества I вычисляется

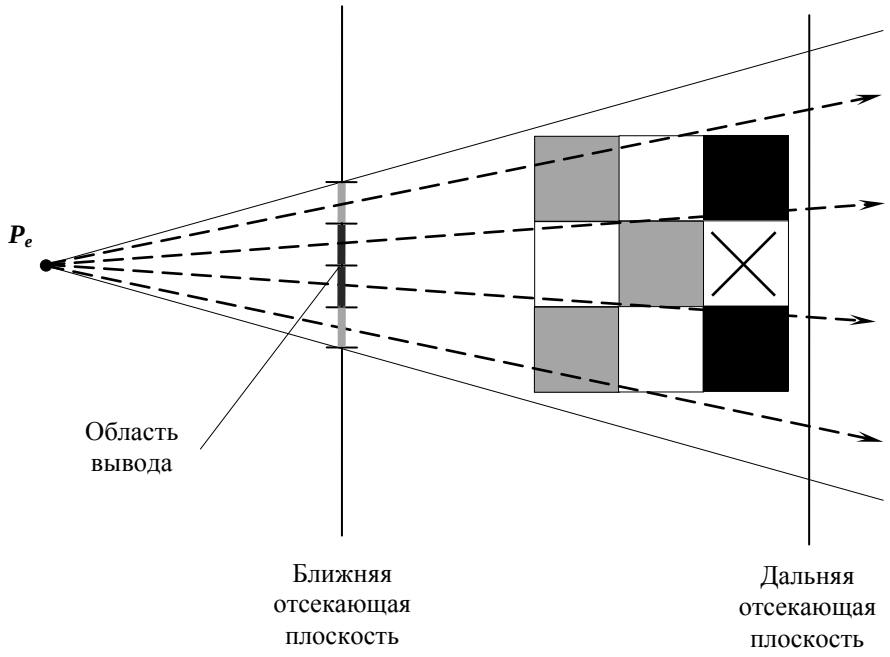


Рис. 2. Ситуация потери значений ячеек скалярного поля

цвет C_{MIP}. Рассмотрим эти этапы более подробно.

2. Метод рендеринга ограничивающего куба *K*

Из рисунка 2 видно: чтобы предотвратить пропуск ячеек скалярного поля F проецирующим лучами, необходимо, чтобы каждую заднюю (по отношению к наблюдателю) ячейку пересекал хотя бы один луч, т.е. сторона ячейки должна быть минимум в 1,5 раза больше, чем проекция стороны пиксела на заднюю грань ограничивающего куба. Обозначим через l_c длину стороны ячейки в системе координат наблюдателя (*View Coordinate System, VCS*) [6], а через l_p - длину проекции стороны пиксела на заднюю грань ограничивающего куба. Тогда для порогового размера d_{thres} ограничивающего куба можно записать следующее выражение

$$d_{thres} = nl_c = \frac{3}{2} nl_p . \qquad (1)$$

Чтобы найти величину l_p из (1), проведем перпендикуляр из точки P_e к ближней плоскости отсечения (см. рисунок 3). Рассмотрим некоторый пиксел, через сторону которого проходит этот перпендикуляр. Обозначим через λ длину стороны этого пиксела, через d_n - расстояние от наблюдателя до ближней плоскости отсечения, через L - расстояние от наблюдателя до задней стороны ограничивающего куба, а через H - высоту области вывода (все величины в системе VCS).

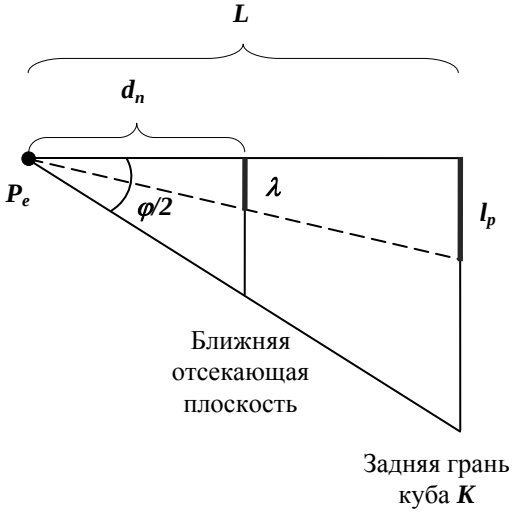


Рис. 3. Проекция пиксела на заднюю грань куба *K*

Из подобия треугольников следует, что

$$l_p = \frac{\lambda}{d_n} L . \qquad (2)$$

Для длины λ стороны пиксела запишем следующее выражение:

$$\lambda = \frac{H}{h},$$

где h - высота области вывода, в пикселах, а величина H находится как (см. рис. 3):

$$H = 2d_n tg\frac{\varphi}{2}.$$

С учетом этого выражение (2) можно записать в следующем виде

$$l_p = \frac{\lambda}{d_n}L = \frac{HL}{d_n h} = \frac{2L}{h}tg\frac{\varphi}{2}. \qquad (3)$$

Подставив (3) в (1) получим искомое выражение для порогового размера d_{thres}:

$$d_{thres} = \frac{3Ln}{h}tg\frac{\varphi}{2}. \qquad (4)$$

Чтобы получить в области вывода изображение ограничивающего куба K (множество I фрагментов), выполним рендеринг смасштабированной в d_{thres} раз полигональной модели единичного куба, заданной лентой (стрипом) треугольников [7]. Вершины модели куба пронумерованы как показано на рисунке 4, а стрип записан в виде следующей последовательности номеров вершин

$$S_K = (0, 1, 3, 2, 6, 1, 5, 0, 4, 3, 7, 6, 4, 5).$$

Исходно координаты вершин модели куба заданы в объектной системе координат (*Object Coordinate System, OCS*), начало которой совпадает с 0-й вершиной куба, а оси направлены вдоль ребер куба (см. рис. 4).

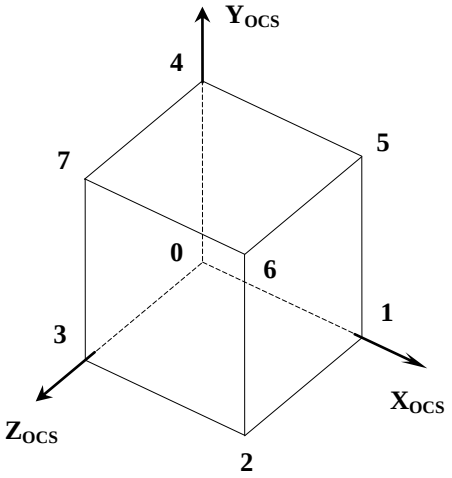


Рис. 4. Полигональная модель ограничивающего куба

Также для каждой вершины куба задан rgb-цвет (его компоненты нормализованы и заданы в вещественном формате), дублирующий координаты этой вершины в системе OCS: 0 - $(0,0,0)$, 1 - $(1,0,0)$, 2 - $(1,0,1)$, 3 - $(0,0,1)$, 4 - $(0,1,0)$, 5 - $(1,1,0)$, 6 - $(1,1,1)$, 7 - $(0,1,1)$. Эти цвета будут использоваться в дальнейшем при расчете координат точек входа и выхода проецирующих лучей из куба K. Цвета и координаты вершин, а также массив S_K индексов помещаются в видеопамять на стадии загрузки системы визуализации с помощью технологии вершинных буферов (*Vertex Buffer Object, VBO*) [8].

Чтобы получить искомое отображение I ограничивающего куба K, мы формируем на GPU с помощью ряда операторов OpenGL модельно-видовую матрицу, преобразующую полигональную модель единичного куба в ограничивающий куб K и выполняем рендеринг полигональной модели куба. Это реализует следующий

Алгоритм рендеринга ограничивающего куба K

1. Зададим размеры области вывода, в которую будет выполняться рендеринг куба K, с помощью функции *glViewport (0, 0, w, h)*.
2. Зададим параметры перспективной проекции с помощью функции

gluPerspective (φ, a, d_n, d_f),

где a - отношение ширины кадра к его высоте, а d_f - расстояние от позиции камеры до дальней отсекающей плоскости.
3. Сформируем текущую модельно-видовую матрицу для рендеринга куба K:
   3.1. Установим единичную модельно-видовую матрицу с помощью функции *glLoadIdentity()*.
   3.2. Вычислим d_{thres} по формуле (4).
   3.3. Сдвинем центр модели куба на расстояние $d_s = L - 0.5d_{thres}$ по направлению $-OZ$ с помощью функции *glTranslatef (0, 0, $-d_s$)*.
   3.4. Выполним масштабирование модели куба на коэффициент d_{thres} с помощью функции *glScalef* (d_{thres}, d_{thres}, d_{thres}).
   3.5. Переместим центр куба в начало мировой системы координат с помощью функции *glTranslatef (-0.5, -0.5, -0.5)*.
4. Визуализируем стрип S_K с помощью технологии VBO.

Конец алгоритма.

Отметим, что в п. 3 алгоритма изменения модельно-видовой матрицы выполняются в порядке, обратном преобразованию полигональной модели куба, т.е. сначала единичный

куб отцентровывается, затем масштабируется и т.д. Для учета случая, когда ограничивающий куб произвольно ориентирован относительно наблюдателя, в пунктах 3.3 и 3.4 вместо d_{thres} можно использовать $d_{thres}\sqrt{3}$ (наибольший размер у куба, когда наблюдатель смотрит вдоль его диагонали).

Приведенный алгоритм реализован в виде функции R рендеринга куба K со следующими значениями: $\alpha = 5.7°$, $a = 16/9$, $d_n = 0.01$, $d_f = 100$, $d_s = 15$.

3. Расчет координат точек выхода проецирующих лучей

В данной работе расчет координат точек выхода проецирующих лучей из куба K выполняется в системе OCS полигональной модели куба. Из рисунка 1 видно, что, с какой бы стороны наблюдатель ни смотрел на куб K, точка P_B выхода луча всегда будет принадлежать некоторому треугольнику задней грани куба (по отношению к наблюдателю). Имея барицентрические координаты точки P_B внутри этого треугольника, можно легко получить координаты точки P_B в системе OCS куба путем интерполяции цветов вершин треугольника (в цветах вершин записаны их координаты в системе OCS, см. раздел 2). Расчет интерполированного по барицентрическим координатам цвета является частью фиксированного функционала графического конвейера (с аппаратной реализацией алгоритма) и выполняется для каждого фрагмента треугольника параллельно и независимо друг от друга в процессе рендеринга.

Определим новую текстуру O размера, совпадающего с размером кадра, в которой r, g и b компоненты хранятся в вещественном формате, а на каждый цветовой канал отводится 32 бита. Чтобы получить координаты точек выхода проецирующих лучей из куба K, выполним рендеринг задних граней куба не в экранный буфер кадра, а в текстурную карту O (внеэкранный рендеринг [6]). Это осуществляет следующий

Алгоритм синтеза карты точек выхода проецирующих лучей из куба K

1. Включим режим отсечения невидимых граней полигонов с помощью функции *glEnable (GL_CULL_FACE)*.
2. Зададим передние грани полигонов в качестве невидимых (отсекаемых) с помощью функции *glCullFace (GL_FRONT)* (по умолчанию в OpenGL невидимыми являются задние грани полигонов).
3. Активируем внеэкранный буфер кадра, связанный с текстурной картой O.
4. Выполним очистку текстуры O цветом $(0,0,0)$ с помощью функции *glClear (GL_COLOR_BUFFER_BIT)*.
5. Выполним рендеринг полигональной модели ограничивающего куба в текстуру O с помощью функции R.
6. Деактивируем внеэкранный буфер кадра.
7. Вернем установку задних граней полигонов в качестве невидимых с помощью функции *glCullFace (GL_BACK)*.

Конец алгоритма.

Отметим, что мы пересоздаем текстуру O только при увеличении размеров окна визуализации, а при уменьшении – размеры текстуры остаются прежними, а уменьшаются только размеры области вывода во внеэкранный буфер кадра.

В результате работы приведенного алгоритма в видеопамяти для текущего кадра визуализации формируются текстурная карта O точек выхода проецирующих лучей из куба K (см. рис. 5).

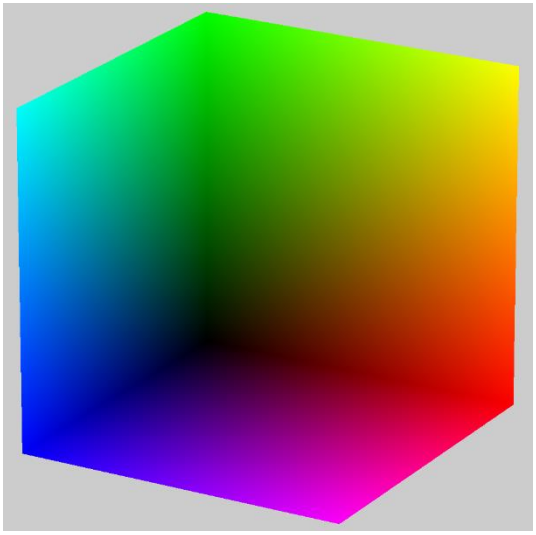


Рис. 5. Синтезированная карта точек выхода проецирующих лучей из куба K

4. Вычисление цветов фрагментов ПМИ

На данном этапе для каждого фрагмента

ПМИ параллельно, независимо друг от друга вычисляется цвет C_{MIP} (см. раздел 1) с помощью разработанного фрагментного шейдера (программы, исполняемой на ядрах GPU).

Данный шейдер осуществляет выборку значений из ячеек скалярного поля F вдоль проецирующего луча; выбор наибольшего значения и расчет для этого значения цвета из 256-ти цветной палитры (цветовая схема палитры – «голубой-синий-пурпурный-красный», где голубой цвет соответствует наименьшему значению скалярного поля, а красный – наибольшему). Для этого в данной работе используется 3D-текстура U (массив 2D-текстур) размера n^3 текселов с одним цветовым каналом (красным), в которой в вещественном 32-битном формате записаны значения скалярного поля F, а также текстурная карта O точек выхода проецирующих лучей (см. раздел 3).

Чтобы построить ПМИ, выполним рендеринг передних граней полигональной модели ограничивающего куба K с помощью функции R и разработанной фрагментной шейдерной программы, которая исполняет следующий

Алгоритм вычисления цвета фрагмента ПМИ

1. Передадим во фрагментный шейдер следующие параметры:
- P_A - координаты точки входа проецирующего луча в куб K (рассчитываются автоматически на основе интерполяции цветов вершин полигональной модели куба K, см. раздел 3);
- n - число ячеек скалярного поля F вдоль ребра ограничивающего куба K;
- n_s - число шагов выборки вдоль проецирующего луча;
- $p_{\max}$ - наибольшее значение скалярного поля F;
- $p_{\min}$ - наименьшее значение скалярного поля F;
- p_{thres} - нижняя граница диапазона визуализируемых значений скалярного поля F.

2. Выполним выборку из текстурной карты O координат P_B точки выхода проецирующего луча с помощью встроенной шейдерной функции считывания текселя (пиксела текстуры):

$$P_B = texelFetch\left(O, P_{screen}\right),$$

где P_{screen} - координаты обрабатываемого фрагмента в экранной системе координат (из встроенной переменной *gl_FragCoord*).

3. Найдем наибольшее значение $f_{\max}$ скалярного поля F вдоль проецирующего луча:

3.1. Запишем координаты и длину l вектора $\vec{r}$ проецирующего луча в системе OCS:

$$\vec{r} = P_B - P_A, \quad l = \left|\vec{r}\right|.$$

3.2. Запишем длину q шага выборки вдоль проецирующего луча:

$$q = l / n_s.$$

3.3. Запишем координаты единичного вектора $\vec{r}_u$ направления проецирующего луча в системе OCS:

$$\vec{r}_u = \vec{r} / l.$$

3.4. Выполним инициализацию $f_{\max}$:

$$f_{\max} = p_{\min}.$$

3.5. Цикл по номеру c шага от 0 до $n_s - 1$

Запишем координаты P_C точки выборки на проецирующем луче:

$$P_C = P_A + qc\vec{r}_u.$$

Запишем тройку индексов (j,i,k) ячейки 3D скалярного поля F, в которой находится точка P_C:

$$j = \lfloor nP_{C,x} \rfloor,\ i = \lfloor nP_{C,y} \rfloor,\ k = \lfloor nP_{C,z} \rfloor.$$

Отсечем значения j,i,k по диапазону $[0, n-1]$.

Выполним выборку значения $f_{j,i,k}$ скалярного поля F из 3D-текстуры U:

$$f_{j,i,k} = texelFetch\left(U, (j,i,k)\right).$$

Обновим значение $f_{\max}$:

$$f_{\max} = \max\left(f_{j,i,k}, f_{\max}\right).$$

Конец цикла.

4. Вычислим цвет C_{MIP} фрагмента ПМИ:

Если $f_{\max} \le p_{thres}$, то $C_{MIP} = (0,1,1,0)$, где четвертая компонента - значение альфа-канала (0 - полностью прозрачный цвет, 1 - полностью непрозрачный).

В противном случае, если $f_{\max} \le \frac{1}{3} p_{\max}$, то $C_{MIP} = (0,1-3t,1,1)$, где $t = \frac{f_{\max}}{p_{\max}}$.

В противном случае, если $f_{\max} > \frac{1}{3} p_{\max}$ и $f_{\max} \le \frac{2}{3} p_{\max}$, то $C_{MIP} = (3t-1,0,1,1)$.

В противном случае $C_{MIP} = (1,0,3-3t,1)$.

Конец алгоритма.

Отметим, что на данном этапе рендеринг

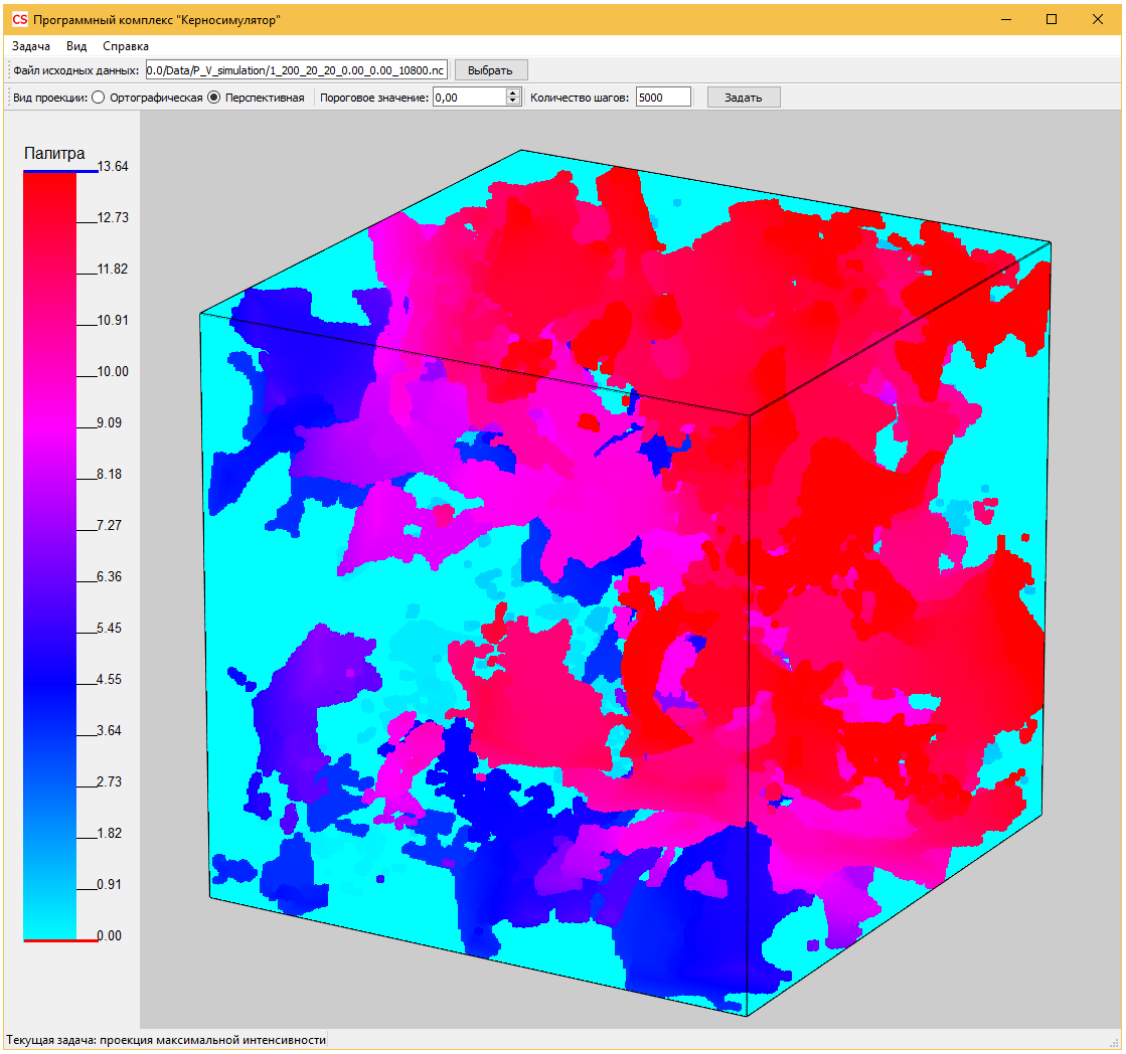


Рис. 6. Модуль визуализации проекции максимальной интенсивности

полигональной модели ограничивающего куба K выполняется со включенным режимом полупрозрачной отрисовки (функция OpenGL *glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA))*, при котором цвета C_{MIP} с 0-й альфа-компонентой не записываются в буфер цвета (в нем остается цвет фона). Вычислив таким образом цвета C_{MIP} всех фрагментов, мы получим в буфере цвета результирующее изображение проекции максимальной интенсивности 3D скалярного поля F.

Заключение

В данной работе рассмотрена задача визуализации проекции максимальной интенсивности 3D скалярного поля, полученного в результате численного эксперимента на цифровой модели керна. Было предложено решение, обеспечивающее построение ПМИ в масштабе реального времени с распараллеливанием графических расчетов на многоядерном GPU. Предложенное решение включает в себя модифицированный метод и алгоритм рендеринга ограничивающего куба 3D скалярного поля, предотвращающий пропуск ячеек поля проецирующими лучами, а также эффективные алгоритмы расчета точек выхода проецирующих лучей из ограничивающего куба и расчета цветов фрагментов ПМИ.

На основе разработанных метода и алгоритмов был реализован программный модуль задачи «Проекция максимальной интенсивности» для комплекса «Керносимулятор» [9], созданного в ФГУ ФНЦ НИИСИ РАН. Была проведена апробация разработанного программного модуля на 3D скалярном поле обезразмеренного давления

(диапазон значений от 0 до 13 единиц, см. рис. 6), полученного на основе цифровой модели керна песчаника размером 200^3 ячеек $(0.59792 \times 0.59792 \times 0.59792$ мм$)$. Визуализация ПМИ данного скалярного поля выполнялась при разрешении Full HD с помощью видеокарты GeForce RTX 2080 (2944 ядра; частота синтеза кадров составила около 100 кадров в секунду. Проведенная апробация подтвердила адекватность предложенных решений поставленной задаче. Полученные результаты могут быть также использованы при создании систем научной визуализации, систем виртуального окружения и др.

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» (0065-2019-0012**).**

Constructing on the GPU maximum intensity projection of 3D scalar fields of digital core material model

P.Yu. Timokhin, K.D. Panteley, E.M. Vozhegov, A.M. Trushin

Abstract: The paper considers the task of visualization of 3D scalar fields obtained as a result of numerical experiments on digital core material models - samples of oilfield rock, digitized using X-ray computed tomography. A modified method and algorithms for constructing the maximum intensity projection (MIP) of a 3D scalar field on a multi-core graphics processor (GPU) are proposed. The modification is aimed at preventing missing the cells during the projecting scalar field by rays. Based on the method and algorithms developed, the "MIP task" program module was implemented and added to "CoreSimulator" software created in SRISA RAS. The module developed was tested on a number of scalar fields obtained as a result of numerical simulation of pressure distribution in digital core material model of the sandstone, which confirmed the adequacy of the proposed solution to the task.

Keywords: visualization, 3D scalar field, maximum intensity projection, digital core material model, real-time, GPU.

Литература

1. В.Б. Бетелин, Н.Н. Смирнов. О проблеме импортонезависимости в нефтегазовой отрасли. Вычислительное моделирование активных воздействий на нефтяные пласты. «Математика и информационные технологии в нефтегазовом комплексе. Материалы V международной конференции. – Сургут, 17-16 мая 2016 г.», Сургут., Изд-во СурГу, 2017, 8-45.
2. М.В. Михайлюк, Д.А. Кононов, Д.М. Логинов. Метод вычисления множества ячеек двумерной квадратной сетки, пересекаемых заданным лучом. «Труды НИИСИ РАН», т. 9 (2019), № 1, 44-48.
3. J. Amanatides, A.A. Woo. A fast voxel traversal algorithm for ray tracing. «Eurographics 87». – 1987, 3-10.
4. J. Díaz, P. Vázquez. Depth-Enhanced Maximum Intensity Projection. «8th IEEE/EG International Symposium on Volume Graphics». – 2010, 93-100.
5. B. Mora, D. S. Ebert. Low-complexity maximum intensity projection. «ACM Transactions on Graphics», v. 24 (2005), № 4, 1392-1416.
6. R. J. Rost, B. Licea-Kane. OpenGL Shading Language (3rd Edition). Boston, USA, Addison Wesley Professional, 2009.
7. М.В. Михайлюк, Д.А. Кононов, Д.М. Логинов. Визуализация порового пространства в цифровой модели керна. «IT-стандарт», 2017, № 3-1 (12), 7-10.
8. OpenGL. Vertex Specification. Vertex Buffer Object. URL: https://www.khronos.org/opengl/wiki/Vertex_Specification#Vertex_Buffer_Object (дата обращения: 28.05.2019).
9. П.Ю. Тимохин, М.В. Михайлюк. Технология создания мультизадачной графической оболочки системы визуализации цифровой модели керна. «Вестник кибернетики», 2018, № 3 (31), 247-254.

Планирование захвата объектов виртуальным антропоморфным роботом с применением инверсной кинематики

Е.В. Страшнов[1], А.В. Мальцев[2]

ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН»
E-mails: [1]strashnov_evg@mail.ru, [2]avmaltcev@mail.ru

Аннотация: В работе рассматривается задача планирования захвата объектов виртуального окружения с помощью антропоморфного робота. Суть этой задачи заключается в том, чтобы определить целевую конфигурацию (положение и ориентацию) кисти руки и пальцев робота, которая обеспечит надежный захват. Для ее решения был разработан алгоритм управления антропоморфным роботом, основанный на применении метода покоординатного спуска для расчета инверсной кинематики и виртуальных датчиков сенсорного очувствления робота. Апробация алгоритма была проведена на примере захвата антропоморфным роботов объектов стандартной формы в рамках комплекса виртуального окружения, разработанного в ФГУ ФНЦ НИИСИ РАН.

Ключевые слова: система виртуального окружения, антропоморфный робот, планирование захвата, инверсная кинематика, датчики.

Введение

В настоящее время актуальным направлением научных исследований является применение антропоморфных роботов в различных областях деятельности человека. Примером служит экстремальная и космическая робототехника [1], которая характеризуется наличием внешних условий, опасных или вредных для здоровья человека. Антропоморфный робот в этих условиях должен выполнять ряд технологических работ, в которых требуется осуществлять захват, манипуляции и перенос объектов различной формы. При этом управление роботом может осуществляться как в копирующем режиме с применением экзоскелета [2] и киберперчаток [3], надеваемых на тело и руки человека, так и в автоматическом режиме на основе заранее подготовленных программ и показаний датчиков. Разработка алгоритмов автоматического управления антропоморфным роботом для выполнения захвата объектов является сложной и важной задачей.

При управлении антропоморфным роботом для выполнения захвата объекта в автоматическом режиме выделяются несколько этапов, в которые входит подведение рабочего органа (кисти руки) к объекту, его захват, манипулирование и перенос. Подведение рабочего органа к объекту относится к классу задач планирования операции захвата объекта, которая заключается в вычислении целевой конфигурации рабочего органа, задающей положение и ориентацию кисти руки и пальцев. Сложность этой задачи состоит в том, что в общем случае существует большое количество возможных допустимых конфигураций. Подходящую из них следует выбирать на основе набора критериев и ограничений, определяющих качество и надежность захвата роботом объекта. Данную задачу более целесообразно решать в рамках систем виртуального окружения. Это, с одной стороны, уменьшает время разработки систем управления, а с другой стороны – позволяет проверить успешность выполнения роботом операции захвата. Примером служит интерактивный симулятор манипуляционных роботов GraspIt! [4], в котором осуществляется визуализация операции захвата объектов для различных типов захватных устройств роботов. Задаче планирования захвата объекта с помощью манипуляционных роботов посвящено ряд работ [5-8]. Среди предлагаемых решений выделяются подходы, в которых конфигурация рабочего органа робота задается с учетом наличия препятствий [5] и семантической информации [6] о разновидностях объектов.

В данной работе предлагается алгоритм управления антропоморфным роботом, с помощью которого осуществляется захват объекта. Для реализации алгоритма была задействована виртуальная модель торсового антропоморфного робота SAR 401 и

виртуальная сцена, содержащая объекты стандартной формы (сфера, параллелепипед и цилиндр). Предлагаемый алгоритм основан на применении метода покоординатного спуска для расчета инверсной кинематики [9-12] и показаний виртуальных датчиков [13] (положения и ориентации). Целевая конфигурация рабочего органа (кисти руки) задается на основе детерминированной информации о расположении и размеров объектов. Разработанный алгоритм был реализован и апробирован в комплексе виртуального окружения, созданном в ФГУ ФНЦ НИИСИ РАН.

1. Виртуальная модель антропоморфного робота

В данной работе задействована виртуальная модель торсового антропоморфного робота SAR 401 (см. рис. 1), состоящая из корпуса, к которому с помощью шарниров крепится “голова” с виртуальной камерой и два манипулятора в виде рук. Робот является стационарным, так как основание его корпуса неподвижно. Торс робота имеет две вращательные степени свободы, что позволяет ему поворачиваться вокруг двух осей на углы φ и ψ. Корпус робота задается системой координат, в которой точка P – ее начало, а векторы $\mathbf{e}^1$, $\mathbf{e}^2$ и $\mathbf{e}^3$ – направления ее осей. Каждый из манипуляторов робота имеет по восемь вращательных степеней свободы: два в плече, по три – в локте и запястье. Опишем манипулятор правой руки. Повороты в его сочленениях задаются углами q_h^i с ограничениями на их величину $q_{h,\min}^i \le q_h^i \le q_{h,\max}^i$, $i = \overline{1,8}$. Чтобы управлять манипулятором, необходимо изменять положение и ориентацию его рабочего органа (кисти руки). Для этого в центре ладони введена система координат F_h, в которой точка P_h – начало системы координат, а векторы $\mathbf{e}_h^1$, $\mathbf{e}_h^2$ и $\mathbf{e}_h^3$ – направления ее осей. При этом вектор $\mathbf{e}_h^1$ направлен перпендикулярно ладони и задает направление захвата. Аналогичным образом определяется манипулятор левой руки.

Рабочим органом (РО) робота является кисть руки (см. рис. 2), содержащая пять пальцев, где большой палец имеет четыре степени свободы, а остальные пальцы – по три

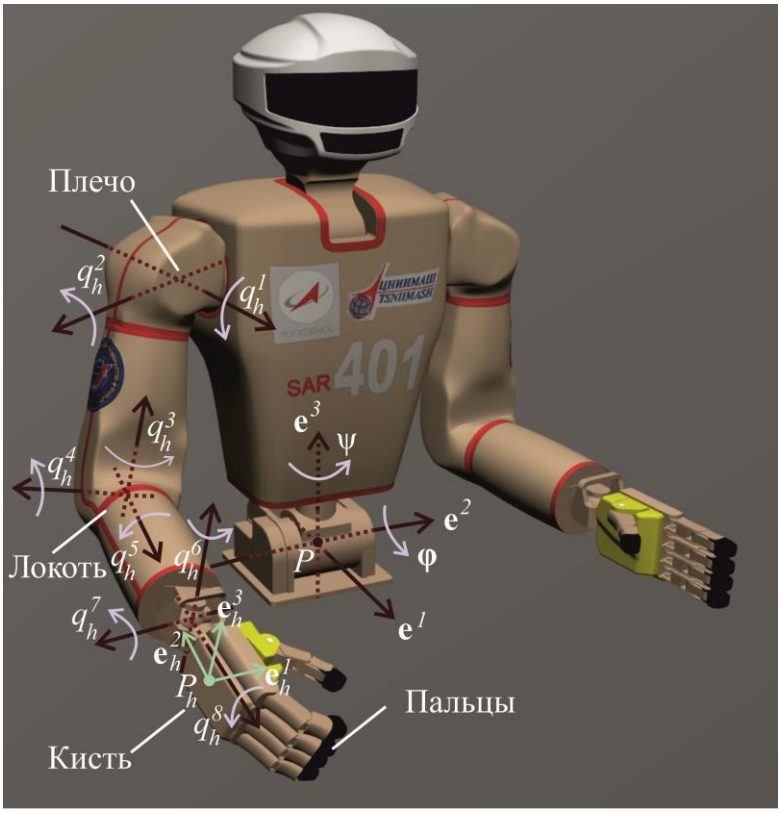


Рис. 1. Виртуальная модель торсового антропоморфного робота

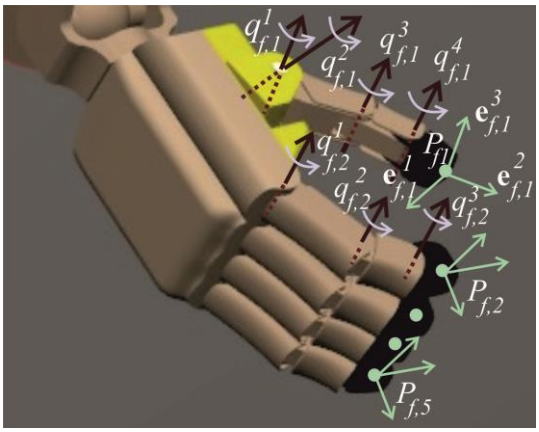


Рис. 2. Кисть руки антропоморфного робота

степени свободы. Повороты в сочленениях пальцев задаются углами $q^j_{f,k}$ с ограничениями на их величину $q^j_{f,k,\min} \le q^j_{f,k} \le q^j_{f,k,\max}$, $j = \overline{1, N_k}$, $k = \overline{1,5}$, где $N_k = 4$ при $k = 1$ и $N_k = 3$ при $k > 1$. Положения и ориентации пальцев задаются с помощью систем координат $F_{f,k}$, в которых точки $P_{f,k}$ – начала систем координат, а векторы $\mathbf{e}^1_{f,k}$, $\mathbf{e}^2_{f,k}$ и $\mathbf{e}^3_{f,k}$ – направления их осей, где векторы $\mathbf{e}^1_{f,k}$ направлены перпендикулярно внутренним плоскостям контакта пальцев.

Для управления антропоморфным роботом в автоматическом режиме необходимо обладать информацией о положениях и ориентациях его составных частей (кистей рук и пальцев). Так как робот является стационарным, то эту информацию можно получить, решая задачу прямой кинематики, в рамках которой системы координат F_h и $F_{f,k}$ могут быть рекуррентно определены через углы q^i_h и $q^j_{f,k}$. Для удобства в рассматриваемой модели антропоморфного робота данная информация вычисляется непосредственно с помощью виртуальных датчиков [13], установленных на роботе и измеряющих положение и ориентацию его составных частей. Датчик положения измеряет координаты x, y и z объекта в мировой системе координат, а датчик ориентации – углы Эйлера с последовательностью поворотов объекта сначала вокруг оси X на угол φ, затем вокруг оси Y на угол ψ и, наконец, вокруг оси Z на угол θ. Для рассматриваемых углов Эйлера матрица поворота имеет вид

$$R = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} = \begin{pmatrix} c_\psi c_\theta & -c_\psi s_\theta & s_\psi \\ c_\varphi s_\theta + s_\varphi s_\psi c_\theta & c_\varphi c_\theta - s_\varphi s_\psi s_\theta & -s_\varphi c_\psi \\ s_\varphi s_\theta - c_\varphi s_\psi c_\theta & s_\varphi c_\theta + c_\varphi s_\psi s_\theta & c_\varphi c_\psi \end{pmatrix},$$

где используются обозначения $c_\alpha = \cos\alpha$ и $s_\alpha = \sin\alpha$.

При выполнении условия $|\psi| < \frac{\pi}{2}$ углы Эйлера могут быть получены из матрицы R следующим образом

$$\begin{aligned} \psi &= \arcsin(m_{13}); \\ \varphi &= \arctan 2(-m_{23}/c_\psi, m_{33}/c_\psi); \\ \theta &= \arctan 2(-m_{12}/c_\psi, m_{11}/c_\psi), \end{aligned} \qquad (1)$$

где функция $\arctan 2(y, x)$ – арктангенс от двух аргументов.

Показания датчиков используются при управлении антропоморфным роботом.

2. Инверсная кинематика

Задача инверсной кинематики (ИК) для антропоморфного робота состоит в том, чтобы определить углы поворотов q^i_h и $q^j_{f,k}$ в сочленениях, при которых РО (кисть руки) манипулятора будет переведен в целевую систему координат F^d_h, а его пальцы – в системы координат $F^d_{f,k}$, где $i = \overline{1,8}$, $j = \overline{1, N_k}$, $k = \overline{1,5}$. Эта задача в общем случае может иметь несколько решений или не иметь их вообще. В случае отсутствия решения необходимо приблизить РО к целевым системам координат насколько это возможно. Мы рассматриваем такое управление манипулятором руки, в котором движение кисти руки и его пальцев выполняется независимо друг от друга.

Для решения задачи ИК в статье задействован итерационный метод покоординатного спуска [11], идея которого заключается в последовательных поворотах шарниров манипулятора таким образом, чтобы уменьшить суммарную ошибку положения и ориентации целевой системы координат. Опишем реализацию этого метода на примере управления РО манипулятора антропоморфного робота. В этом случае задача состоит в том, чтобы найти изменения

углов Δq_h^i поворотов в шарнирах, обеспечивающие максимум целевой функции

$$g_h(\Delta q_h^i) = w_p P_h^d \cdot R(\mathbf{n}_h^i, \Delta q_h^i) P_h^c + w_o \sum_{s=1}^{3} \sigma_s \mathbf{e}_h^{s,d} \cdot \mathbf{e}_h^{s,c}(\Delta q_h^i), \; i = \overline{1,8}, \quad (2)$$

где w_p и w_o – весовые коэффициенты; σ_s – коэффициенты учета ориентации; P_h^c и P_h^d – текущее и целевое положение РО; $R(\mathbf{n}_h^i, \Delta q_h^i)$ – матрица поворота вокруг оси $\mathbf{n}_h^i$ i-го шарнира на угол Δq_h^i; $\mathbf{e}_h^{s,d}$ и $\mathbf{e}_h^{s,c}(\Delta q_h^i)$ – векторы, задающие оси целевой ориентации РО и его текущей ориентации после поворота на угол Δq_h^i.

Максимум целевой функции (2) достигается, когда выполнено условие $\frac{dg_h}{d\Delta q_h^i} = 0$, что соответствует уравнению относительно Δq_h^i.

Решая это уравнение, получим

$$\Delta q_h^i = \arctan 2(k_3, k_2 - k_1), \quad (3)$$

где коэффициенты k_1, k_2 и k_3 вычисляются как

$$k_1 = w_p (P_h^d \cdot \mathbf{n}_h^i)(P_h^c \cdot \mathbf{n}_h^i) + w_o \sum_{s=1}^{3} \sigma_s (\mathbf{e}_h^{s,d} \cdot \mathbf{n}_h^i)(\mathbf{e}_h^{s,c} \cdot \mathbf{n}_h^i);$$

$$k_2 = w_p (P_h^d \cdot P_h^c) + w_o \sum_{s=1}^{3} \sigma_s (\mathbf{e}_h^{s,d} \cdot \mathbf{e}_h^{s,c});$$

$$k_3 = \mathbf{n}_h^i \left(w_p (P_h^d \times P_h^c) + w_o \sum_{s=1}^{3} \sigma_s (\mathbf{e}_h^{s,d} \times \mathbf{e}_h^{s,c}) \right).$$

В алгоритме метода покоординатного спуска применяется итеративный обратный проход [9] шарниров, начиная от шарнира, ближайшего к РО.

Для каждого шарнира решается уравнение (3) и полученное значение угла Δq_h^i добавляется в общий угол q_h^i, для которого проверяются ограничения $q_{h,\min}^i \le q_h^i \le q_{h,\max}^i$.

Если эти ограничения нарушаются, то угол Δq_h^i корректируется таким образом, чтобы они были выполнены.

Метод является итеративным и реализуется в несколько проходов. Выбирая определенным образом весовые коэффициенты w_p, w_o и σ_s, можно управлять его скоростью сходимости.

Аналогичным образом формулируется и реализуется ИК для управления движением пальцев РО.

3. Планирование захвата объектов

Под задачей планирования здесь понимается получение такой конфигурации РО (кисти руки и пальцев), при которой обеспечивается надежный захват объекта антропоморфным роботом. Для решения этой задачи мы рассматриваем полностью детерминированные виртуальные сцены, в которых положения, ориентации и размеры объектов заранее известны. Объектами являются геометрические примитивы (сфера, параллелепипед и цилиндр), которые изначально располагаются на плоской поверхности (столе) и имеют такие размеры, что позволяет им помещаться внутри кисти робота. В данной работе мы ограничиваемся рассмотрением захватов, в которых вектор $\mathbf{e}_h^1$ кисти руки параллелен плоскости стола. В этом случае целевая конфигурация захвата определяется положением кисти руки P_h^d и его пальцев $P_{f,k}^d$, а ориентация – углами θ_h^d и $\theta_{f,k}^d$ поворотов кисти руки и пальцев вокруг оси Z, перпендикулярной плоскости стола.

Рассмотрим этапы обеспечения целевой конфигурации захвата объекта антропоморфным роботом.

На первом этапе корпус робота поворачивается на углы φ_d и ψ_d по направлению к центру объекта. Эти углы могут быть вычислены как

$$\varphi_d = -\arcsin\left(\frac{\mathbf{P}_c\mathbf{P}'}{\|\mathbf{P}_c\mathbf{P}'\|} \cdot \mathbf{e}^3\right); \quad (4)$$

$$\psi_d = \arcsin\left(\frac{\mathbf{P}_c\mathbf{P}}{\|\mathbf{P}_c\mathbf{P}\|} \cdot \mathbf{e}^2\right),$$

где точка P_c является центром объекта, а точка $P' = P + \Delta z \mathbf{e}^3$ задает вертикальное положение руки манипулятора на величину Δz относительно плоскости стола.

На втором этапе осуществляются повороты в сочленениях пальцев на углы $\tilde{q}_{f,k}^j$ так, чтобы полностью раскрыть захват робота.

На третьем этапе решается задача ИК и вычисляются углы $q_h^{i,(1)}$ поворотов манипулятора руки, перемещающие кисть руки в точку $P_h^{d,(1)}$ с ориентацией, задаваемой углом θ_h^d. При этом, чтобы манипулятор не столкнулся с объектами в процессе его перемещения, целевая точка $P_h^{d,(1)}$ кисти руки выбирается на заданном расстоянии над

объектом. В свою очередь угол θ_h^d подбирается на основе текущей ориентации объекта (для параллелепипеда) и с учетом ориентации корпуса робота после поворота на угол ψ_d.

На четвертом этапе решается задача ИК для углов $q_h^{i,(2)}$, перемещающих кисть руки в точку $P_h^{d,(2)}$ таким образом, чтобы она находилась на определенном расстоянии справа от объекта, а проекции пальцев попадали на него.

Наконец, на пятом этапе, подбирая эвристическим образом положения $P_{f,k}^d$ и ориентации $\theta_{f,k}^d$, осуществляется сжатие захвата. Для этого решается задача ИК и вычисляются углы $q_{f,k}^j$ поворотов пальцев. При этом целевые точки $P_{f,k}^d$ пальцев выбираются такими, что они проникают на небольшое расстояние внутрь объекта. Это сделано для того, чтобы в процессе захвата полностью затянуть объект внутрь кисти руки, образуя тем самым контакт пальцев и ладони с объектом.

На всех этапах достижение точек и углов своих целевых значений проверяется на основе выполнения критериев малости для расстояний и разности углов.

Алгоритм планирования захвата объектов на основе рассмотренных этапов будет следующим:

1. Поворачиваем корпус робота на углы φ_d и ψ_d по направлению к центру объекта P_c, согласно соотношениям (4).

2. Раскрываем захватное устройство – поворачиваем пальцы робота на углы $\tilde{q}_{f,k}^j$.

3. Цикл, пока не будут выполнены условия $\left\|\mathbf{P}_h^{d,(1)}\mathbf{P}_h^c\right\| \le \varepsilon_1$ и $\left|\theta_h^d - \theta_h^c\right| \le \varepsilon_2$:

- решаем задачу ИК – вычисление углов $q_h^{i,(1)}$ для целевого положения $P_h^{d,(1)}$ и угла θ_h^d кисти руки робота.

4. Цикл, пока не будут выполнены условия $\left\|\mathbf{P}_h^{d,(2)}\mathbf{P}_h^c\right\| \le \varepsilon_1$ и $\left|\theta_h^d - \theta_h^c\right| \le \varepsilon_2$:

- решаем задачу ИК – вычисление углов $q_h^{i,(2)}$ для целевого положения $P_h^{d,(2)}$ и угла θ_h^d кисти руки робота.

5. Цикл, пока не будут выполнены условия $\left\|\mathbf{P}_{f,k}^d\mathbf{P}_{f,k}^c\right\| \le \varepsilon_1$ и $\left|\theta_{f,k}^d - \theta_{f,k}^c\right| \le \varepsilon_2$:

- решаем задачу ИК – вычисление углов $q_{f,k}^j$ для целевых положений $P_{f,k}^d$ и углов $\theta_{f,k}^d$ пальцев руки робота.

Конец алгоритма.

В этом алгоритме точки P_h^c и $P_{f,k}^c$ – текущие положения кисти руки и пальцев, а углы θ_h^c и $\theta_{f,k}^c$ – их повороты, вычисляемые из соотношений (1) для угла θ. Величины $\varepsilon_1 > 0$ и $\varepsilon_2 > 0$ задают точность критериев достижения целевых точек и углов.

4. Результаты моделирования

Предлагаемый алгоритм планирования захвата объекта антропоморфным роботом был реализован в комплексе виртуального окружения, созданном в ФГУ ФНЦ НИИСИ РАН. Этот программный комплекс состоит из

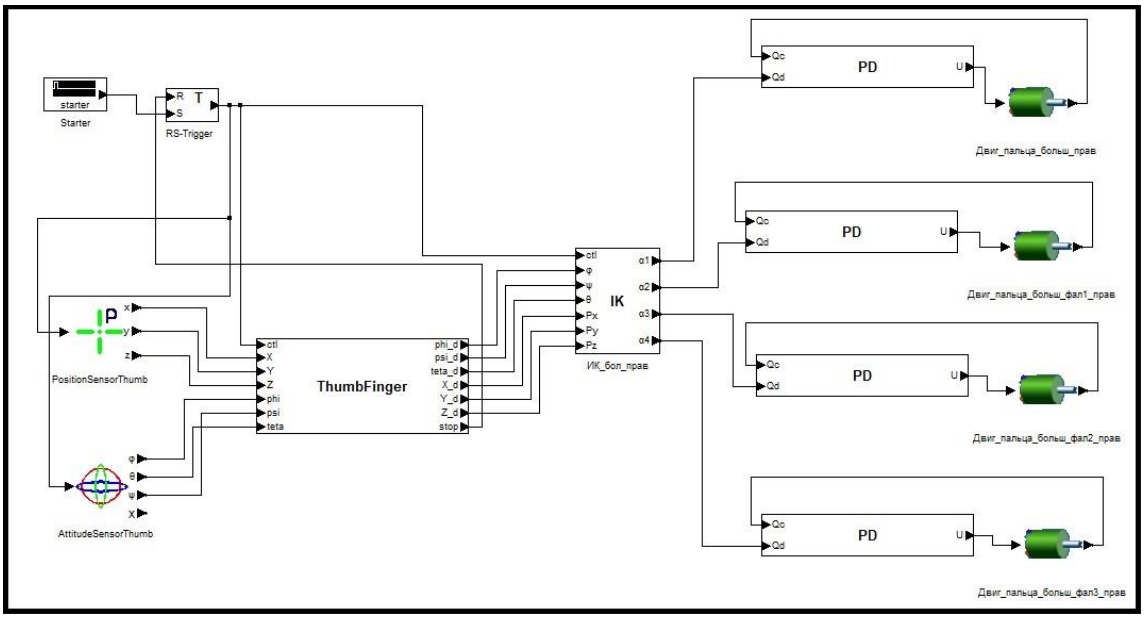


Рис. 3. Функциональная схема управления антропоморфным роботом

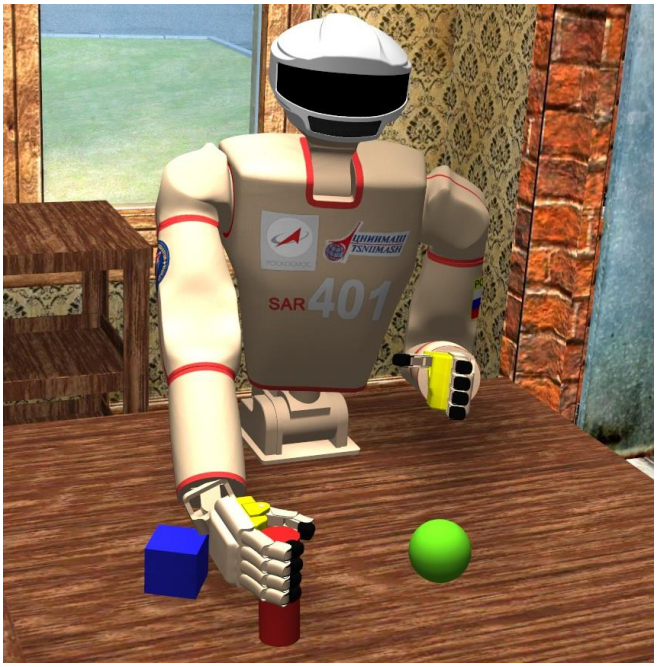


Рис. 4. Захват цилиндра виртуальным антропоморфным роботом

подсистем динамики, управления и визуализации. Подсистема управления вычисляет управляющие воздействия, подаваемые на исполнительные устройства робота. Эти воздействия передаются в подсистему динамики, в которой вычисляются новые координаты объектов, определяются показания датчиков, а также выполняется расчет инверсной кинематики. Вычисленные координаты объектов передаются в подсистему визуализации, которая осуществляет рендеринг виртуальной сцены.

Управление антропоморфным роботом в рассматриваемом программном комплексе осуществляется с помощью функциональной схемы, которая состоит из блоков датчиков, исполнительных устройств, инверсной кинематики, ПД-регуляторов, а также блоков библиотеки редактора функциональных схем. На рисунке 3 показан фрагмент тестовой схемы для управления большим пальцем руки антропоморфного робота. В этой схеме блок “Starter” формирует единичный импульс в начале моделирования, подаваемый на вход S блока “RS-Trigger”, переключая его в логическую единицу. Блоки датчиков “PositionSensorThumb” и “AttitudeSensorThumb” вычисляют текущее положение и углы Эйлера ориентации большого пальца, которые передаются на вход блока “ThumbFinger”. В этом блоке формируется целевое положение $P_{f,1}^{d}$ и ориентация $\theta_{f,1}^{d}$, а также выполняется сравнение с показаниями датчиков $P_{f,1}^{c}$ и $\theta_{f,1}^{c}$. Целевые координаты передаются на вход блока инверсной кинематики, в котором с помощью метода покоординатного спуска вычисляются углы поворотов $q_{f,1}^{j}$, $j=\overline{1,4}$ в сочленениях большого пальца. Эти углы передаются на входы блоков ПД-регуляторов [9], которые вычисляют напряжения, подаваемые на входы электродвигателей. Рассматриваемая схема работает до тех пор, пока на выходе stop блока “ThumbFinger” не возникнет единица, означающая достижение целевых координат. Эта единица передается на вход R блока “RS-Trigger”, переключая его в логический ноль.

Моделирование проводилось в виртуальной сцене, содержащей антропоморфный робот и три объекта разного типа (сфера, параллелепипед и цилиндр). На рисунке 4 показано конечное состояние руки манипулятора после успешного захвата цилиндра.

Заключение

В работе была рассмотрена задача планирования автоматического захвата объекта антропоморфным роботом в системах виртуального окружения. Для ее решения был разработан алгоритм, в котором для управления положением и ориентацией кисти

руки и пальцев был реализован расчет инверсной кинематики с помощью метода покоординатного спуска. При этом для проверки достижения целевых координат были задействованы показания виртуальных датчиков положения и ориентации. Апробация разработанного алгоритма для планирования захвата объекта была проведена в полностью детерминированной виртуальной сцене, содержащей объекты стандартной формы. В дальнейшем предполагается развитие методов и подходов управления антропоморфным роботом с применением датчика силы и момента для решения задачи удержания объекта.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 17-07-00137.

Grasp planning for virtual anthropomorphic robot using inverse kinematics

E.V. Strashnov, A.V. Maltsev

Abstract: This paper deals with the task of grasp planning of virtual environment objects by an anthropomorphic robot. The challenge is to determine the target configuration (position and orientation) of the robot hand and its fingers, ensuring robust grasping. To solve this problem, an algorithm for anthropomorphic robot control was developed, based on the use of the coordinate descent method for inverse kinematics and readings of virtual sensors. The approbation of proposed algorithm for anthropomorphic robot grasping of geometric primitives was carried out in the virtual environment complex developed in SRISA RAS.

Keywords: virtual environment system, anthropomorphic robot, grasp planning, inverse kinematics, sensors.

Литература

1. Е.И. Юревич. Основы робототехники. 2-е изд., перераб. и доп. Спб., БХВ-Петербург, 2005.
2. М.А. Торгашев. Моделирование копирующего режима управления антропоморфным роботом в виртуальной среде. «Труды НИИСИ РАН», т. 5 (2015), № 2, 47-54.
3. C.W. Borst, A.P. Indugula. Realistic virtual grasping. «IEEE Virtual Reality», 2005.
4. A.T. Miller, and P.K. Allen. GrapIt!: A versatile simulator for grasping analysis. «Proc. of the ASME Dynamic Systems and Control Division», vol. 2 (2000), 1251-1258.
5. A.T. Miller, S. Knoop, P.K. Allen and H.I. Christensen. Automatic grasp planning using shape primitives. «Proc. of the Intl. Conf. on Robotics and Automation (ICML)», 2003.
6. Z. Xue, A. Kasper, JM Zollner and R. Dillmann. An automatic grasp planning system for service robots. «14th international conference on advanced robotics (ICAR'09), Munich, Germany, 22-26 June 2009», 1-6.
7. C. Ferrari and J. Canny. Planning optimal grasps. «Proc. of the 1992 IEEE Intl. Conf. on Robotics and Automation», 1992, 2290-2295.
8. C. Borst, M. Fischer and G. Hirzinger. Efficient and precise grasp planning for real world objects. «Springer Tracts in Advanced Robotics», vol. 18 (2005), 91-111.
9. Е.В. Страшнов, М.В. Михайлюк. Моделирование полуавтоматического режима управления манипуляционными роботами в системах виртуального окружения. «Вестник кибернетики», (2017), № 4, 191-198.
10. Е.В. Страшнов, М.А. Торгашев. Супервизорное управление антропоморфными роботами с применением инверсной кинематики. «Виртуальное моделирование, прототипирование и промышленный дизайн: Материалы IV Международной научно-практической конференции, 2017 (тезисы)», Тамбов, Издательский центр ФГБОУ ВО «ТГТУ», вып. 4, т. 1 (2017), 186-190.
11. Li-Chun Tommy Wang and Chih Cheng Chen. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. «IEEE Transactions on Robotics and Automation», vol. 7 (1991), no. 4, 489-499.
12. B. Kenwright. Inverse kinematics – cyclic coordinate descent (ccd). «Journal of Graphics Tools», vol. 16 (2012), no. 4, 177-217.
13. М.В. Михайлюк, Е.В. Страшнов, Д.М. Логинов. Моделирование датчиков в системах виртуального окружения. «Труды НИИСИ РАН», т. 8 (2018), № 2, 70-76.

Алгоритм компактного документирования видеоданных эндоскопии трубовидных объектов

В.К. Салахутдинов[1], А.В. Голубкин[1], Д. Дорошенко[2], О.С. Стрельцова[3], Е.А. Монакова[1], А.Г. Прозорова[1], А.А. Меркулова[1]

[1] НИИ системных исследований РАН
[2] ООО "Центр диагностики и хирургии заднего отдела глаза"
[3] ФГБОУ ВО "ПИМУ" Минздрава России

E-mail's: vsalakhutdinov@gmail.com, alexandr_go@bk.ru, retina@retina.ru, strelzova_uro@mail.ru, lenamonakova@gmail.com, aleksashka.prozorova@yandex.ru, enfantter@gmail.com

Аннотация. Представлены результаты разработки методов и средств обработки и компактного документирования данных телевизионной эндоскопии. Предложена схема регистрации эндоскопических изображений. Разработан метод устранения пространственной ошибки модуляции. Показано, что при полном сохранении разрешения и качества цветопередачи предложенный метод позволяет сократить объем хранимых данных более чем на три порядка. Экспериментальные результаты показывают эффективность предложенного метода.

Ключевые слова. Телевизионная эндоскопия, регистрация эндоскопических изображений, устранение пространственной ошибки модуляции.

Введение

Согласно прогнозам развития цифровой медицины [1, 6] уже в ближайшее время видеорегистрация всей совокупности эндоскопических манипуляций может стать одним из обязательных требований страховой медицины к высокотехнологичным медицинским услугам. Предполагается, что это позволит доказательно, путем инструментального сравнения состояния пациента до, в процессе и после лечения, обосновать целесообразность применения высокотехнологичной медицинской услуги. Проблема в том, что эндоскопические видеоданные плохо поддаются компрессии и поэтому для их хранения требуется неоправданно большой объем памяти. Другая проблема на пути широкого применения видеодокументирования состоит в том, что подавляющее большинство исследуемых средствами эндоскопии органов (кровеносные сосуды, элементы желудочно-кишечного тракта и т.д.), имеют трубовидную топологию [2]. Как следствие, регистрируемое с помощью эндоскопа изображение представляет собой совокупность плохо поддающихся сшивке двусвязных фигур (колец) [3]. Из-за этого эффективное использование результатов видеорегистрации требует применения принципиально новых средств распознавания, классификации и поиска.

В работе представлены результаты разработки нового метода обработки и компактного документирования данных видеорегистрация объектов с трубовидной топологией.

При этом в первой части представлено формальное описание задачи и нового алгоритма ее решения, а во второй описан алгоритм устранения ошибок паразитной пространственной модуляции. Результат эксперимента представлен в заключительной части.

Идея предлагаемого подхода заключается в реализации алгоритма конформного отображения непригодного для сшивки (двусвязного) регистрируемого эндоскопом изображения в виде кольца на пригодную для сшивки односвязную прямоугольную область.

1. Алгоритм компактного документирования

Структурная схема видеорегистрации представлена на рисунке 1 и состоит из трубовидного объекта, изображение внутренней стенки которого регистрируется видеокамерой эндоскопа, расположенной внутри трубовидного объекта.

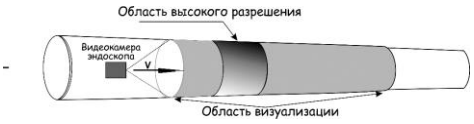


Рис.1. Структурная схема видеорегистрации

Видеокамера двигается вдоль трубовидного объекта со скоростью V_0 и регистрирует изображение его внутренней стенки с частотой кадров n_i, удовлетворяющей условию $n_{i>>}R_0^{-1} V_0$, где R_0 – радиус кривизны внутренней стенки. Пример одиночного видеокадра, регистрируемого расположенным внутри мочеточника эндоскопом, представлен на рисунке 2. Видно, что регистрируемое изображение представляет собой кольцо, внутри которого масштаб уменьшается по мере удаления от центра, большая (заштрихованная) часть которого неинформативна из-за сильной дифокусировки и аберраций.

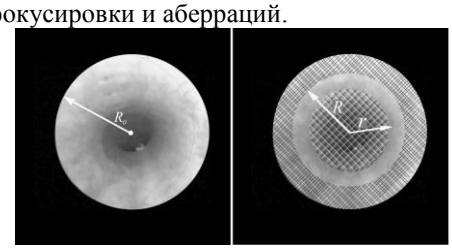
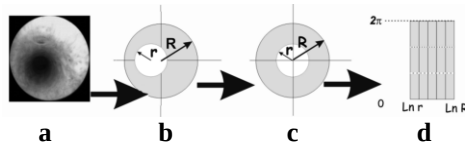


Рис.2 Видеокадр регистрируемого изображения

Также видно, что информационно значимая область представляет собой кольцо с внешним радиусом R и внутренним радиусом r.

Визуализация полной картины обследуемого объекта требует сшивки информационно значимых областей всех видеокадров, зарегистрированных видеокамерой эндоскопа в процессе движения. Алгоритм сшивки показан на Рис.3. На Рис.3a представлен исходный видеокадр. Преобразование из 3b в 3c трансформирует информационно значимую область в центр и устраняет искажения масштаба, а преобразование 3c в 3d трансформирует двусвязную информационно значимую область в виде кольца в односвязное изображение в виде прямоугольника.



a **b** **c** **d**

Рис.3 Структурная схема алгоритма сшивки

Очевидно, что преобразование из 3b в 3c тривиально реализуется с помощью дробно-линейного конформного преобразования вида:

$$z_1 = \frac{az + b}{cz + d}.$$

Для этого достаточно определить его значения в трех точках. В качестве таковых выбираем остающиеся неподвижными две точки на действительной оси – (+R) и (–R), и центр внутреннего круга:

$$z_1^1 = z_1(-l) = 0,\ z_1^2 = z_1(-R) = -R,\ z_1^3 = z_1(R) = R.$$

Подставляя эти значения в формулу, определяющую параметры дробно-линейного преобразования по трем точкам, получаем уравнение

$$\frac{z_1}{z_1 + R} \cdot \frac{2R}{R} = \frac{z + l}{z + R} \cdot \frac{2R}{R + l}.$$

Решение которого имеет вид:

$$z_1 = \frac{z + l}{(zl / R^2) + 1}$$

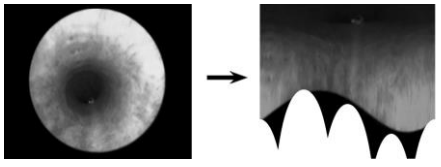
Преобразование 3c в 3d, трансформирующее двусвязную информационно значимую область в виде кольца в односвязное изображение в виде прямоугольника, может быть реализовано функцией вида $ln\ z$.

Таким образом окончательный вид искомого конформного преобразования из 3a в 3d может быть реализован в виде композиции двух полученных преобразований вида:

$$w = \ln \frac{z + l}{(zl / R^2) + 1}$$

Очевидно, что последующая сшивка прямоугольных фрагментов в общее изображение представляет собой простую техническую задачу.

На рис.4 представлен результат эксперимента по сшивке серии изображений с помощью описанного алгоритма.



a b

Рис. 4. Пример развертки эндоскопического изображения; на 4a – кадр исходное изображение, 4b – результат сшивки

Видно, что угловое и пространственное смещение оптической оси видеокамеры относительно оси обследуемого объекта и вариации скорости движения эндоскопа приводят к паразитной пространственной модуляции результирующего изображения.

2.Алгоритм устранения паразитной пространственной модуляции

Для устранения паразитной пространственной модуляции

Рассмотрим решение в следующем порядке по параметру 1/R, т.е. будем считать, что у трубы для одного снимка (ограничивающего небольшой участок поверхности вдоль трубы) есть большой (но конечный) постоянный радиус кривизны R. Этот радиус кривизны меняется при существенном перемещении видеокамеры. Однако на двух соседних кадрах видеоряда его можно считать примерно постоянным.

Если для всех точек одного изображения радиус кривизны постоянен, то трехмерные координаты этих точек лежат на поверхности тора (см. рис. 5), следовательно, удовлетворяют уравнениям:

$$\begin{cases} x(\varphi,\psi) = (R + r\cos\varphi)\cos\psi, \\ y(\varphi,\psi) = (R + r\cos\varphi)\sin\psi, \\ z(\varphi,\psi) = r\sin\varphi. \end{cases} \qquad (1)$$

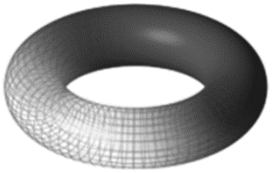


Рис. 5. Участок трубы, аппроксимированный тором заданного большого радиуса R

Для каждого снимка необходимо определить направление на центр тора (т.е. вектор вдоль большого радиуса кривизны). Это можно сделать, например, считая, что оптическая ось видеокамеры расположена вдоль тора, и направление на центр тора на двумерном изображении соответствует отрезку, соединяющему центр оптической оси объектива и центр центральной темной области изображения рис. 6.

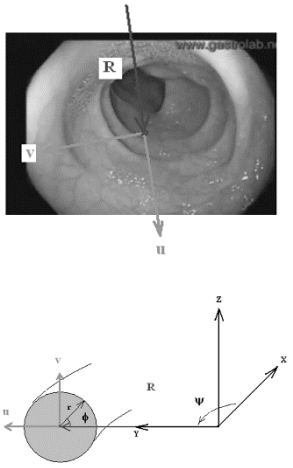


Рис. 6. Выбор системы координат в каждом участке снимка

Выберем систему координат *(u, v)* в плоскости апертуры видеокамеры ориентированную так, что ось ***v*** направлена вдоль **z**, u – вдоль y, а ось x перпендикулярна *(u, v)*. Координаты точки на изображении (*u, v*) связаны с трехмерными координатами на торе соотношениями перспективной проекции вида:

$$y = fz/x,\ u = f(Y-R)/x, \qquad (2)$$

где *f* – фокусное расстояние камеры.

Предположим теперь, что видеокамера перемещается вдоль канала. При перемещении между двумя соседними кадрами считатаем, что видеокамера остается неподвижной, а тор поворачивается навстречу видеокамере, т. е. координата ψ для заданной точки на поверхности тора изменяется на малый угол $d\ \psi$. При этом остальные координаты (r и φ) остаются неизменными. При таком повороте координаты *x, y, z* изменяются на малые величины, а пространственные смещения описываются уравнениями:

$$\begin{cases} dx = -(R + r\cos\varphi)\sin\psi\ \ d\psi = -(R + r\cos\phi)d\psi, \\ dy = (R + r\cos\varphi)\cos\psi\ \ d\psi = 0, \\ dz = 0 \end{cases}$$

Поскольку для выбранного положения системы координат $\psi = \pi/2$.

Подставляя это в (2), получим:

$$dv = -fzdx/x^2,\ \ du = fRdx/x^2,$$

откуда:

$$dv/du = z/R. \qquad (3)$$

Из (3) видно, что искажения устраняются применением

следующего алгоритма:

Определяем координату *z*, исходя их предыдущих формул для прямой трубы, смещение одинаковых участков изображения определяем с помощью алгоритма SIFT [4] для поиска парных соответствий на изображениях (см. рис. 6). Это позволяет определить величины *dv/du* для указанной точки, и из соотношения (3) определяем радиус кривизны канала. Далее, действуя итеративно, находим примерный радиус кривизны канала для каждого положения зонда. Это дает возможность восстановить форму канала от его начала и пространственное положение видеокамеры в нем.

Результат реконструкции изображения фрагмента мочеточника [5] длиной ~10 см. – представлен на рис.7. Размер исходного видеоряда цветных изображений - более 400 Мбайт. При этом размер реконструированного изображения составил менее 3 Мбайт.

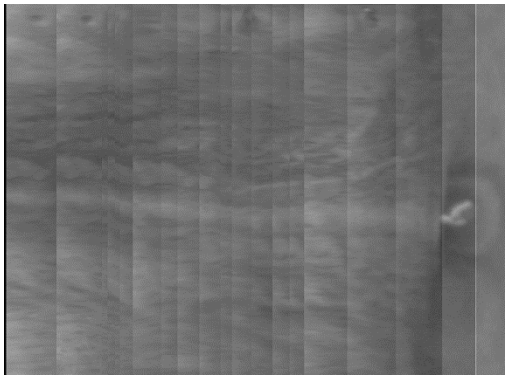


Рис. 7. Реконструированное эндоизображение 10 см фрагмента мочеточника

Видно, что при полном сохранении разрешения и качества изображения предложенный метод позволяет сократить объем регистрируемых данных более чем на два порядка.

Исследования выполнены при частичном финансировании в рамках проектов РФФИ № 19-07-00395 и № 18-07-00882.

Algorithm of compact documentation of video data of endoscopy of tubular objects

V.K. Salakhutdinov, A.V. Golubkin, D. Doroshenko, O.S. Streltsova, E.A. Monakova, A.G. Prozorova, A.A. Merkulova

Abstract. Methods and tools for processing and documenting TV endoscopy data are presented. A structural scheme for TV image registration is proposed. A method for correcting the spatial error of modulation is developed. The method was shown to reduce the volume of the data to store more than by three orders of magnitude while retaining the resolution and the quality of color of the transmitted images. Experimental results demonstrate the efficiency of the proposed method.

Key words. TV endoscopy, registration of endoscopic images, elimination of spatial modulation error.

Литература

1 Маады А.С., Алексеев К.И., Осипов А.С., Васильев И.В Инновационные эндоскопические технологии в многопрофильном медицинском учреждении «Вестник Национального медико-хирургического Центра им. Н.И. Пирогова», т. 12 (2017), № 4.

2 Murat Akarsu, MD and Cevher Akarsu, MD, Evaluation of New Technologies in Gastrointestinal Endoscopy, JSLS, V.22(1) (2018).

3 Кирсанов И.И., Гуляев А.А., Пахомова Г.В., Ярцев П.А., Левитский В.Д., Черныш О.А. Видеолапароскопия при прободной язве желудка и двенадцатиперстной кишки «Эндоскопическая хирургия», т. 16 (2010), № 1, 8-12.

4 Oh J.-H., Hwang S., Tavanapong W., de Groen P.C., Wong J. Blurry-frame detection and shot segmentation in colonoscopy videos. «IS&T/SPIE Symposium on Electronic Imaging 2004», San Jose, CA, USA, SPIE, 2004, 531-542.

5 Chen Ying-ju, Yasen W., Jeongkyu L.E.E., Dongha L.E.E., Yongho K.I.M. Developing assessment system for wireless capsule endoscopy videos based on event detection. «Proc SPIE», V. 7260 (2009), 1-11.

6 Dudenkova V.V., Maslennikova A.V., Kiseleva E.B., Tararova E.A., Yunusova K.E., Streltsova O.S. Quantitative Assessment of Radiation-Induced Changes in the Connective Tissue Matrix of the Urinary Bladder by Nonlinear Microscopy. «Sovremennye tehnologii v medicine», V.10 (2018), № 3, 118–124.

Анализ механизмов распределения прибыли в модели прозрачной экономики

З.Б. Сохова

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, *E-mail: zarema.sokhova@gmail.com*

Аннотация. В статье рассматривается взаимодействие инвесторов и производителей в прозрачной экономической системе. Анализируются механизмы распределения прибыли производителями и влияние этих механизмов на решения, которые принимаются инвесторами. Модель исследована с помощью компьютерного моделирования, проведен анализ полученных результатов.

Ключевые слова: агент-ориентированная модель, автономные агенты, инвесторы, производители, прозрачная экономика.

Введение

В настоящее время для анализа и исследования поведения сложных систем часто применяются методы агент-ориентированного моделирования (АОМ) [1, 2]. АОМ широко используется при решении задач оптимизации и управления, моделировании коллективного поведения, в социальных [3] и экономических исследованиях [4].

Описав агентов и простые правила их поведения (микроскопический уровень), мы имеем возможность наблюдать за тем, как функционирует система в целом (макроскопический уровень). Отметим, что даже если правила поведения агентов достаточно просты, вся система в целом может демонстрировать сложное поведение.

Данная работа продолжает работы начатые в [5, 6, 7] по исследованию поведения агентов-инвесторов и агентов-производителей в прозрачной экономической среде. В этой статье изучаются возможные механизмы распределения прибыли производителями, рассматриваются две модели: 1) без учета собственного вклада производителя (*базовая модель*) и 2) с учетом собственного вклада (*новая модель*). Ниже будет дано подробное описание каждого подхода. Каждый агент в модели принимает решение самостоятельно, при этом, благодаря возможности сотрудничества (обмен информацией и прозрачность среды), возможно более эффективное функционирование всей системы в целом.

1. Краткое описание базовой модели

Опишем сначала кратко базовую модель, предложенную в [5, 6]. Рассматривается взаимодействие двух сообществ агентов: *агентов-инвесторов* и *агентов-производителей*. Число инвесторов равно N, число производителей M, их капиталы равны: K_{inv} (инвесторы) и K_{pro} (производители). Агенты функционируют в течение N_T периодов. В каждом периоде T инвесторы должны решить какие вклады они будут делать в производителей в следующем периоде $T+1$. Для того чтобы принять решение, организуется итеративный процесс, в течение которого инвесторы и производители открыто обмениваются информацией. Количество итераций внутри периода равно t_{max}. Для обмена информацией используются легкие агенты: *агенты-разведчики* и *агенты намерения*, аналогичные тем, которые были использованы в [8].

В начале периода T i-й производитель располагает капиталом C_i, в котором учтены вклады инвесторов:

$$C_i = C_{i0} + \sum_{j=1}^{N} C_{ij}, \qquad (1)$$

где C_{i0} – собственный капитал производителя, C_{ij} – капитал, вложенный j-м инвестором в i-го производителя в начале периода. Прибыль i-го производителя определяется по формуле $P_i(C_i) = k_i F(C_i)$, где функция F одинакова для всех производителей, коэффициент k_i –

эффективность производства i-го производителя. Функция $F(x)$ имеет вид:

$$F(x) = \frac{x^2}{x^2 + a^2} \ , \qquad (2)$$

где a – положительный параметр.

После получения прибыли производитель выплачивает ее часть инвесторам. При этом j-му инвестору отдается часть прибыли, пропорциональная сделанному им вкладу в данного производителя:

$$P_{inv\ ij} = k_{repay} P_i(C_i) \frac{C_{ij}}{\sum_{l=1}^{N} C_{il}} \ , \qquad (3)$$

где C_i – текущий капитал (в начале периода) i-го производителя, k_{repay} – параметр, характеризующий долю прибыли, которая выплачивается инвесторам, $0 < k_{repay} < 1$. Заметим, что в этом случае, при распределении прибыли производители не учитывают размер собственного вклада C_{i0}, а отдают часть прибыли согласно параметру k_{repay} (см. выражение (3)), определяющему долю прибыли, которую отдельный производитель выплачивает инвестору. Сам производитель получит:

$$P_{pro\ i} = P_i(C_i) - \sum_{j=1}^{N} P_{inv\ ij} \ . \qquad (4)$$

Инвесторы определяют вклады в течение итеративного процесса. При этом важно, то, что инвесторы при принятии решения о вкладах *принимают во внимание действия других инвесторов*. Подробно этот итеративный процесс изложен в статье [9] (см. также текст статьи на сайте: https://niisi.ru/tr/2018_T8_N2.pdf).

В модели учитывается амортизация и инфляция. Для этого капиталы производителей в конце каждого периода T пересчитываются с учетом амортизации $K_{pro}(T+1) = k_{amr} K_{pro}(T)$, где k_{amr} – коэффициент амортизации $(0 < k_{amr} \le 1)$. Аналогично пересчитываются капиталы инвесторов $K_{inv}(T+1) = k_{inf}\, K_{inv}(T)$, где k_{inf} – коэффициент инфляции $(0 < k_{inf} \le 1)$.

Агенты могут делиться, если их капитал превысил определенный порог $Th_{max\ inv}$ (инвесторы) или Th_{max_pro} (производители). При этом численность агентов в сообществе должна быть меньше максимально возможной. При делении «агент-родитель» отдает половину своего капитала «потомку». Производитель-«родитель» передает по наследству свою эффективность k_i.

Если же капитал агента становится меньше определенного малого порога $Th_{min\ pro}$ (производители) или Th_{min_inv} (инвесторы), то такой агент погибает.

2. Новая модель

В описанной выше базовой модели при формировании оценок инвесторами и распределении прибыли производителями не учитывается вклад самого производителя. Независимо от того какой собственный вклад C_{i0} был у производителя, полученная отдельным производителем прибыль распределяется между производителем и инвесторами согласно параметру выплат k_{repay}. В новой модели будем учитывать в оценках, которые формируют инвесторы и при распределении прибыли производителями вклады самого производителя (т.е. его собственный капитал C_{i0}). При этом прибыль инвестора определяется выражением (модифицируем формулу (3)):

$$P_{inv\,ij} = Pr_i(C_i) \frac{C_{ij}}{\sum_{l=1}^{N} C_{il} + C_{i0}} \qquad (5)$$

А вклады инвесторов в производителей определяются с помощью итеративного процесса, аналогичного изложенному в работе [9]. В результате при распределении прибыли каждый агент (и производитель, и инвестор) получит прибыль, пропорциональную сделанному им вкладу.

3. Результаты моделирования

В данном разделе представлены результаты моделирования для 1) базовой модели и 2) новой модели.

Были выбраны следующие параметры моделирования: число периодов $N_T = 1$ или 100; максимальное число итераций внутри периода $t_{max} = 40$; максимальные пороги капиталов для инвесторов и производителей $Th_{max\_inv} = 1.0$, $Th_{max\_pro} = 1.0$; минимальные пороги капиталов для инвесторов и производителей $Th_{min\_inv} = 0.01$, $Th_{min\_pro}=0.01$; максимально возможное число производителей и инвесторов в сообществе $M_{max} = 2$ или 100 $N_{max} = 1$ или 100; начальное число производителей и инвесторов $M_0 = 2$ или 100, $N_0 = 1$ или 100; число производителей в которых делаются вклады $m = 2$ или 100; параметр функции прибыли $a = 1$ или 10; коэффициент выплат $k_{repay} = 0.5$; коэффициенты амортизации и инфляции $k_{amr} = 1.0$, $k_{inf} = 1.0$; характерная величина вариации коэффициента эффективности производителей $\Delta k = 0.01$.

Для более четкого представления влияния механизма, выбранного производителем, на то, как распределяет капитал отдельный инвестор, проведено моделирование для частного случая одного инвестора и двух производителей со следующими параметрами:

- эффективности производителей: $k_1 = 0.34$, $k_2 = 0.94$;
- капитал инвестора $K_{inv} = 0.54$;
- капиталы производителей: $K_{pro1} = 0.48$, $K_{pro2} = 0.26$.

На рис. 1 показано как инвестор *перераспределяет* капитал в зависимости от номера итерации в двух режимах работы.

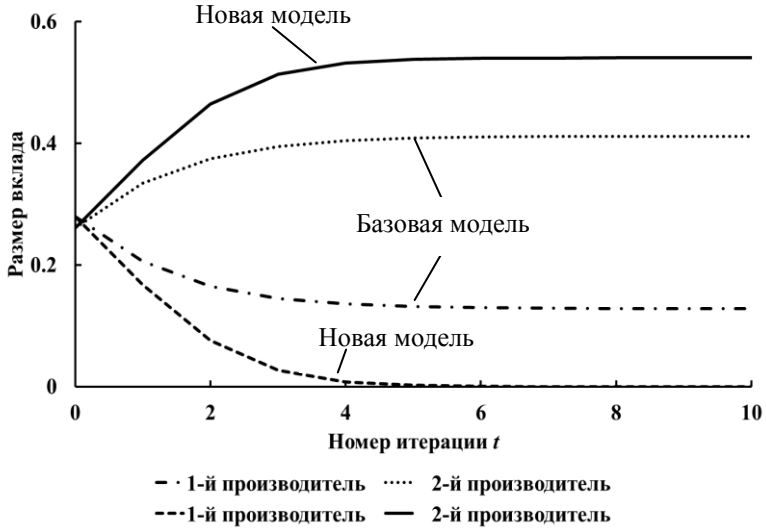


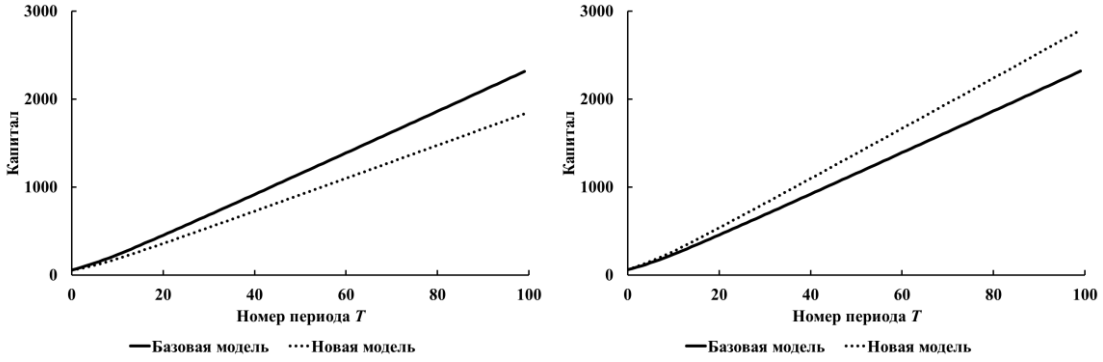
Рис. 1. Распределение вклада одного инвестора в зависимости от номера итерации в периоде $T = 0$, $a = 1$

При сравнении результатов моделирования (рис.1), видно, что в базовой модели инвестор делает вклады в двух производителей, а в новой модели инвестор выбирает только одного, наиболее эффективного производителя, несмотря на то что его капитал ниже.

Далее рассмотрим случай большого сообщества, когда $N = M = 100$. Результаты моделирования для двух моделей представлены ниже на рис. 2.

При анализе результатов в этом случае получаем, что, в базовой модели, когда производители отдают половину прибыли инвесторам, суммарная прибыль сообщества производителей больше (рис. 2(*а*)).

С другой стороны, для инвесторов выгоднее новая модель (рис. 2(*б*)), так как по результатам моделирования в этом режиме суммарный капитал сообщества инвесторов выше.



а) *Производители* *б*) *Инвесторы*

Рис. 2. Динамика суммарных капиталов инвесторов и производителей для базовой и новой модели при $N = M = 100$, $a = 1$

Проведено также моделирование при параметре a = 10. В этом случае прибыль производителей растет медленнее. Результаты представлены на рис. 3. Видно, что в новой модели суммарный капитал сообщества производителей значительно меньше, чем, суммарный капитал производителей в базовой модели. При этом в базовой модели сообщества инвесторов и производителей развиваются одинаково, так как прибыль делится между сообществами поровну (k_{repay} = 0.5).

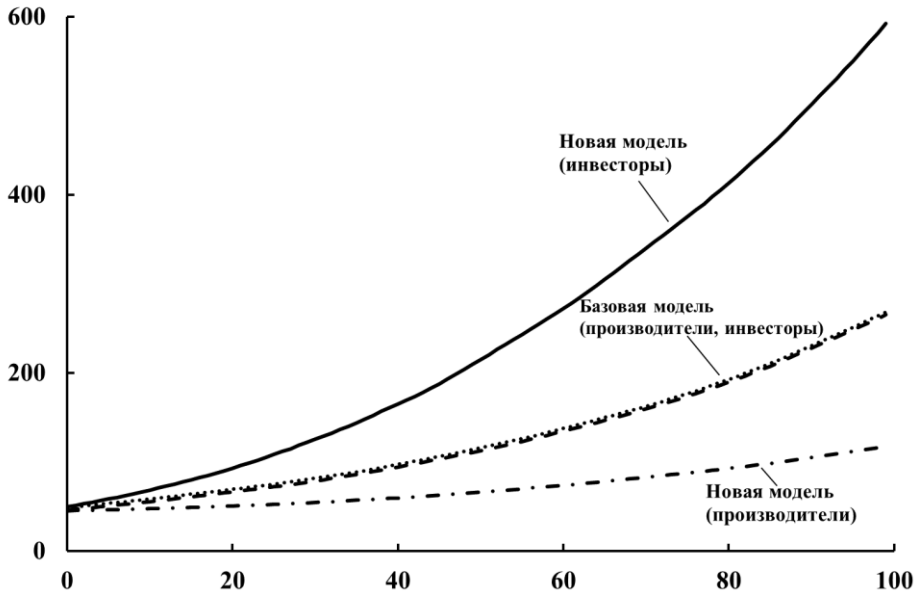


Рис. 3. Динамика суммарных капиталов инвесторов и производителей для базовой и новой модели при $N = M$ =100, a = 10

Учитывая результаты экспериментов, можно сделать вывод о том, что производителям выгоднее выбирать механизм распределения прибыли, предложенный в базовой модели, т.е. не учитывать при распределении прибыли собственный вклад, а делать выплаты прибыли инвесторам согласно коэффициенту выплат k_{repay}. Можно предположить, что в базовой модели более «справедливое» сотрудничество у сообществ.

Выводы

Таким образом, в работе кратко изложена базовая модель взаимодействия инвесторов и производителей в среде прозрачной экономической системы. Важной особенностью модели является открытость информации о намерениях и характеристиках агентов и сотрудничество между агентами.

Предложена новая модель распределения прибыли производителями. Проведено сравнение двух моделей: *базовой* и *новой*. Показано, что механизм, предложенный в базовой модели более выгоден для сообщества производителей, а механизм, предложенный в новой модели, предпочтителен для инвесторов, но при этом сообщество производителей получает слабое развитие.

Данная работа выполнена при финансовой поддержке РФФИ грант № 19-01-00331 «Обобщение, обучение и эволюция в моделях автономных когнитивных агентов»

Analysis of profit distribution mechanisms in a transparent economy model

Z.B. Sokhova

Abstract. The article discusses the interaction of investors and producers in a transparent economic system. The mechanisms of profit distribution by producers and the influence of these mechanisms on decisions that are made by investors are analyzed. The model was investigated using computer simulation, the analysis of the results was carried out.

Keywords: agent-based model, autonomous agents, investors, producers, transparent economy.

Литература

1. N. Gilbert. Agent-based models. Sage Publications, Inc, 2007.
2. E. Bonabeau. Agent-based modeling: methods and techniques for simulating human systems. «Proceedings National Academy of Sciences», v. 99 (2002), 7280 – 7287.
3. R. Axtell. Why agents? On the varied motivations for agent computing in the social sciences. «Center for Socialand Economic Dynamics Working Paper», (2000), №17, 1 − 23.
4. L. Tesfatsion. Agent-Based Computational Economics: A Constructive Approach to Economic Theory. «Handbook of Computational Economics. Agent-Based Computational Economics», v. 2 (2006), 831 − 880.
5. V.G. Red'ko, Z.B. Sokhova. Model of Collective Behavior of Investors and Producers in Decentralized Economic System. «Procedia Computer Science», v. 123 (2018), 380 – 385.
6. V.G. Red'ko, Z.B. Sokhova. Iterative Method for Distribution of Capital in Transparent Economic System. «Optical Memory & Neural Networks (Information Optics)», v. 26 (2017), №3, 182 − 191.
7. В.Г. Редько, З.Б. Сохова. Модель взаимодействия инвесторов и производителей в прозрачной экономической системе. «Экономика и математические методы», т. 54 (2018), № 2, 50 – 61.
8. 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), № 2, 364 – 373.
9. З.Б. Сохова, В.Г. Редько. Исследование процессов самоорганизации в эволюционной модели прозрачной децентрализованной экономики. «Труды НИИСИ РАН», т. 8 (2018), № 2, 103 – 110.