Векторизация алгоритмов анализа динамического напряженно-деформированного состояния для архитектур x86 с поддержкой векторных регистров

В.В. Гетманский, А.Е. Андреев, Е.С. Харьков, Е.О. Мовчан

Волгоградский государственный технический университет

При решении задач моделирования объектов, представленных системой многих тел, в частности, при анализе динамического напряженно-деформированного состояния требуется выполнять большое число матричных и векторных операций с матрицами небольшой размерности на каждом шаге интегрирования систем дифференциальных уравнений. В работе рассматривается сокращение времени выполнения таких операции под архитектуры процессоров x86 с использованием систем команд SSE, AVX, AVX2, AVX512 для центральных процессоров Intel® Xeon, сопроцессоров Intel® Xeon Phi™ (KNC), а также центральных процессоров Intel® Xeon Phi™ второго поколения (KNL).

Ключевые слова: оптимизация кода, векторизация, векторные регистры, динамическое напряженно-деформированное состояние, AVX-512, KNC, KNL

1. Введение

Для расчета динамических напряжений традиционно применяют квазистатические подходы метода конечных элементов. Альтернативный метод дискретных элементов [1], который не требует решения системы линейных уравнений и использует ортогональную сетку, позволяет анализировать напряжения в динамике и при этом эффективно векторизовать и распараллеливать расчетный код.

Разрабатываемый расчетный модуль является частью пакета моделирования многотельной динамики ФРУНД [2], в котором используются связанные задачи анализа динамики конструкции и физических процессов напряжений и теплопередачи в отдельных ее деталях.

Достижение максимальной производительности при решении реальных прикладных задач предполагает использование всех возможностей архитектур современных процессоров и часто требует (как и в синтетических тестах) векторизации вычислений, причем в ряде случаев ручная векторизация оказывается более эффективной по сравнению с оптимизацией, выполняемой компиляторами. Дополнительный интерес к вопросам векторизации вычислений вызван появлением архитектур MIC (Intel Xeon Phi) первого, а затем второго поколения и системы команд AVX512 с поддержкой векторных регистров. В последние годы появилось много публикаций, посвященных оптимизации решателей в различных областях для архитектуры Xeon Phi. В частности, в работе [3] авторы рассматривают подходы к получению максимальной производительности на кластере, оснащенном Xeon Phi, и отмечают необходимость векторизации для дальнейшего роста производительности, в работе [4] автор использует автоматическую векторизацию и достигает за счет этого существенного прироста производительности. В работе [5] авторы описывают, в том числе, способы векторной оптимизации выполнения операций с небольшими матрицами и блоками векторов в рамках многоуровневой реализации параллелизма при обработке разреженных матриц, часто встречающейся в системах инженерного анализа.

Предыдущие исследования авторов показали возможность получения ускорения в рассматриваемом решателе за счет векторизации под различные векторные регистры [6-8]. В данной работе описаны полученные результаты для новых процессоров Intel Xeon Phi второго поколения (KNL) с набором 512-битных регистров (AVX-512).

2. Метод дискретных элементов

Оптимизируется код решателя, выполняющего расчет напряженно-деформированного состояния системы тел со связями. Подобные решатели систем многотельной динамики нашли широкое применение при проектировании и инженерном анализе механических конструкций, где они используются, к примеру, для определения оптимальной массы деталей, жесткости соединительных элементов, оптимальной геометрической формы конструкции в целом.

Динамический анализ позволяет оценить характеристики движения всей механической системы в целом, для его проведения используется метод моделирования динамики системы абсолютно твёрдых тел со связями. В таком подходе в качестве основной модели используется модель динамики системы тел со связями, а вспомогательные модели описывают физические процессы в отдельных телах. Особенности расчета в решателе рассмотрены в предыдущих работах [3-5]. Задача моделируется системой обыкновенных дифференциальных уравнений второго порядка. Расчет правых частей уравнений, который является наиболее вычислительно сложным, осуществляется в несколько шагов. Сначала вычисляется матрица поворота для перевода дискретного элемента в систему координат опорного тела по формуле:

$$\mathbf{A}_{21} = \mathbf{A}_{02}^{\mathrm{T}}\,\mathbf{A}_{01},$$

где первая цифра индекса означает номер системы координат, в которую необходимо перевести положение тела, вторая цифра – текущую систему координат. Основу вычислений составляют операции над матрицами и векторами:

$$\mathbf{d}_{\mathrm{sl}}^{(1)} = \mathbf{C}_1^{(1)} - \mathbf{A}_{21}^{\mathrm{T}}\mathbf{C}_2^{(2)} - \mathbf{A}_{01}^{\mathrm{T}}(\mathbf{P}_2 - \mathbf{P}_1)^{(0)}, \quad (1)$$

$$\mathbf{d}_{vl}^{(1)} = \mathbf{W}_1^{(1)} \times \mathbf{C}_1^{(1)} - \mathbf{A}_{21}^{\mathrm{T}}(\mathbf{W}_2^{(2)} \times \mathbf{C}_2^{(2)}) - \mathbf{A}_{01}^{\mathrm{T}}(\mathbf{V}_2 - \mathbf{V}_1)^{(0)}, \quad (2)$$

$$\mathbf{d}_{sa}^{(0)} = \mathbf{R}_1^{(0)} - \mathbf{R}_2^{(0)}, \quad (3)$$

$$\mathbf{d}_{va}^{(0)} = \mathbf{W}_1^{(0)} - \mathbf{W}_2^{(0)}, \quad (4)$$

где $\mathbf{C}_1^{(1)}$ и $\mathbf{C}_2^{(2)}$ – вектора точек связи, записанные в локальной системе координат первого и второго тела (верхний индекс), $\mathbf{P}_1^{(0)}$ и $\mathbf{P}_2^{(0)}$ – координаты центров масс дискретных элементов в глобальной системе координат, $\mathbf{W}_1^{(1)}$ и $\mathbf{W}_2^{(2)}$ – угловые скорости, $\mathbf{V}_1^{(0)}$ и $\mathbf{V}_2^{(0)}$ – линейные скорости дискретных элементов.

Выражение (1) задает линейный сдвиг дискретного элемента, (2) – изменение линейной скорости, (3) – угловой сдвиг и (4) – изменение угловой скорости. На основе данных выражений вычисляются итоговые значения деформаций, с помощью которых обновляются данные о текущем состоянии модели – пересчет линейных и угловых компонент, обновление сил и моментов и др. По обновленным параметрам выполняется очередной шаг интегрирования явным методом Рунге-Кутты 4-го порядка точности. Основной вклад во время вычислений вносит расчет правых частей, расчет и применение матриц поворота [4].

3. Векторизация решателя динамического напряженно-деформированного состояния с помощью 512-битных регистров

Из формул (1-4) видно, что основные вычисления представляют собой матрично-векторные операции. Векторизация расчетного кода проводится с предварительным выравниванием памяти и имеет ряд особенностей для 512-битных инструкций. При имеющейся линейной структуре хранения данных без изменения алгоритма при выполнении матричных операций используются только 4 элемента двойной точности для каждой строки матрицы или вектора. Как известно, регистры KNC способны вместить 8 чисел типа double, а это значит, что половина каждого регистра в ходе расчета не используется, что существенно снижает производительность программы. В связи с этим возникает необходимость модификации алгоритма перемножения матриц для обработки одновременно нескольких пар матриц и векторов, например, как представлено на рис. 1.

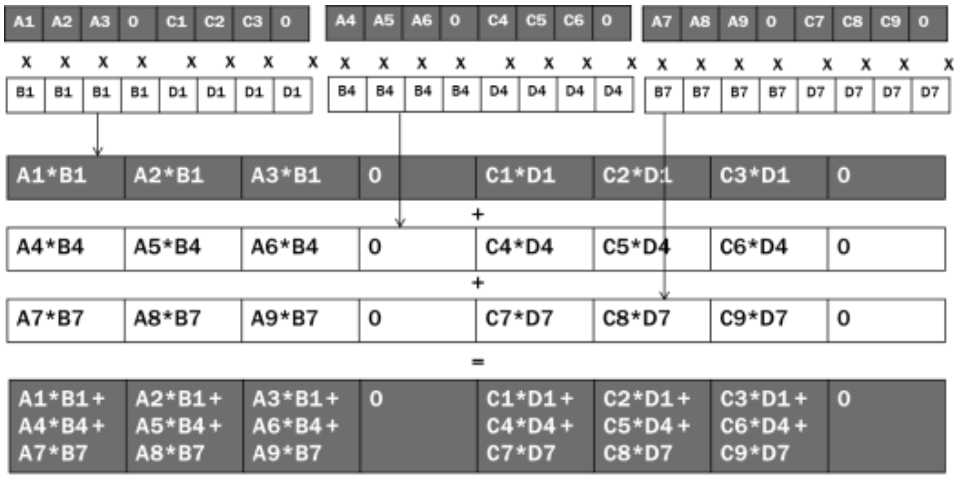

Рис. 1. Примерная схема векторизованного перемножения матриц с помощью регистров KNC

Один из подходов к решению данной проблемы предложен ранее в литературе, в нем желаемое содержимое регистров получается путем последовательных операций размножения и смешивания элементов внутри векторного регистра:

```
    zmm08=_mm512_extload_ps(vec+ 0, _MM_UPCONV_PS_NONE, _MM_BROADCAST_1X16,
_MM_HINT_NONE);
    zmm09=_mm512_extload_ps(vec+ 4, _MM_UPCONV_PS_NONE, _MM_BROADCAST_1X16,
_MM_HINT_NONE);
    zmm10=_mm512_extload_ps(vec+ 8, _MM_UPCONV_PS_NONE, _MM_BROADCAST_1X16,
_MM_HINT_NONE);
    zmm11=_mm512_extload_ps(vec+ 12, _MM_UPCONV_PS_NONE, _MM_BROADCAST_1X16,
_MM_HINT_NONE);


    zmm06 = _mm512_mask_blend_ps(mask32_0, zmm08, zmm09);
    zmm07 = _mm512_mask_blend_ps(mask32_1, zmm10, zmm11);
    zmm04 = _mm512_mask_blend_ps(mask32_2, zmm06, zmm07);
```

Рассмотрен также еще один способ организации пересылок данных между регистрами, поддерживаемый векторным расширением KNC, с помощью команды перестановки и размножения элементов в регистре swizzle:

```
    zmm04 = _mm512_swizzle_ps(zmm08, _MM_SWIZ_REG_AAAA);
    zmm05 = _mm512_swizzle_ps(zmm08, _MM_SWIZ_REG_BBBB);
    zmm06 = _mm512_swizzle_ps(zmm08, _MM_SWIZ_REG_CCCC);
    zmm07 = _mm512_swizzle_ps(zmm08, _MM_SWIZ_REG_DDDD);
```

Очевидно, в данном случае второй вариант является более предпочтительным, так как он позволяет получить желаемый результат за меньшее число шагов и, следовательно, получить больший выигрыш по времени. Авторами был реализован вариант алгоритма векторизации, использующий всю ширину векторных регистров для KNC.

4. Сравнение результатов оптимизации с помощью 512-битных регистров по сравнению с автовекторизацией

В современных компиляторах при включенной оптимизации происходит автоматическая оптимизация кода. Основная операция оптимизации, дающая существенное ускорение – это автовекторизация. Для решателя выделены 3 этапа – расчет поворотов, расчет правых частей и численное интегрирование. Для расчета поворотом проведена оптимизация вычисления тригонометрических функций с помощью инструкций Intel SVML, что ускорило их в 2.5 раза. Рассмотрим на типовой модели пластины из 4000 дискретных элементов (размерностью 200x200) эффективность векторизации расчетного кода.

Таблица 1. Результаты выполнения расчета на одном ядре процессора Intel Xeon E5 с FMA-инструкциями

1 ядро	XEON auto, t,c	XEON FMA, t,c	Ускорение
Повороты	1,69	1,69	1,00
правые части	33,4	15,33	2,18
Интегрирование	0,65	0,44	1,48
Всего	35,74	17,46	2,05

В таблице 1 представлены данные о времени выполнения расчета с одной и той же моделью с автовекторизацией (auto) и с ручной векторизацией с применением FMA инструкций на процессоре Intel Xeon E5. Общее ускорение вычислений за счет ручной векторизации - в 2 раза.

В таблице 2 представлены данные о времени выполнения расчета с одной и той же моделью с автовекторизацией (auto) и с ручной векторизацией с применением KNC инструкций (FMA) на процессоре Intel Xeon Phi. Достигнуто общее ускорение за счет ручной векторизации в 1.63 раза, хотя правые части ускоряются лучше. В столбце ”KNC AVX512” замеры с оптимизацией, использующей половину регистров. В столбце ”KNC AVX512-2” – результаты для улучшенного алгоритма с упаковкой строк 2 матриц в один регистр. Стоит отметить, что узким

Таблица 2. Результаты выполнения расчета на одном ядре процессора Intel Xeon PHI с инструкциями KNC

1 ядро	KNC auto, t, c	KNC AVX512, t, c	KNC AVX512-2, t, c	Ускорение
повороты	337,22	339,11	338,25	1,00
правые части	438,1	196,44	139,32	3,14
интегрирование	10,6	3,42	3,51	3,02
Всего	785,92	538,97	481,08	1,63

местом является расчет поворотов (он является более вычислительно затратным в данном случае). Расчет правых частей ускоряется лучше, чем на CPU (Xeon E5) – более чем в 3 раза. Однако по общему времени вычислений одно ядро Phi уступает одному ядру CPU в 28 раз.

Таблица 3. Результаты выполнения расчета на одном ядре процессора Intel Xeon PHI с инструкциями KNL

1 ядро	KNL auto, t, c	KNL FMA, t, c	KNL AVX-512, t, c	Ускорение
Повороты	7,4	7,4	7,4	1,00
правые части	131,9	61,7	52,9	2,49
Интегрирование	3,77	0,76	0,61	6,18
Всего	143,07	69,86	60,91	2,35

Процессор с архитектурой KNL использует все предыдущие наборы инструкций x86-архитектуры и AVX-512. В таблице 3 показаны результаты времени расчетов на процессоре Intel KNL (7210). За счет ручной векторизации достигается общее ускорение в 2.35 раза. В столбце ”KNL AVX512” приведены замеры с оптимизацией, использующей 512-битные регистры, в столбце ”KNL FMA” – результаты для FMA-версии оптимизированного кода, запущенного на KNL. Одно ядро KNL уступает ядру CPU (Xeon E5) в 3.5 раза, что гораздо лучше, чем в случае с KNC.

5. Распараллеливание векторизованного кода

В архитектуре рассматриваемых процессоров блок векторных регистров содержится в каждом ядре, поэтому для архитектур MIC (many integrated cores), к которым относятся KNC и KNL, имеет смысл рассматривать только выполнение параллельного кода. Код распараллелен с помощью OpenMP с применением распараллеливания циклов.

Для начала рассмотрим выполнение параллельного расчета на процессоре Xeon E5. Результаты приведены в таблице 4. Из них следует, что если сравнивать результат выполнения автовекторизованного кода на одном ядре с кодом, векторизованным вручную на 10 ядрах, общий эффект от оптимизации - это ускорение всего расчета в 11 раз.

Таблица 4. Результаты выполнения расчета на одном ядре и на всех ядрах процессора Intel Xeon E5

все ядра/потоки	XEON auto 10/1, t, c	XEON FMA 10/10, t, c	XEON, ускорение
повороты	1,69	1,11	1,52
правые части	33,4	2	16,70
интегрирование	0,65	0,06	10,83
всего	35,74	3,17	11,27

Следующий результат получен для KNC и приведен в таблице 5.

Таблица 5. Результаты выполнения расчета на одном ядре и на всех ядрах процессора Intel Xeon Phi

все ядра/потоки	KNC auto 60/1, t, c	KNC AVX512 60/240, ускорение	KNC AVX512 60/60, ускорение
повороты	337,22	21,8	18,01
правые части	438,1	6,52	7,27
интегрирование	10,6	0,09	0,13
всего	785,92	28,41	25,41

Из приведенных результатов следует, что если сравнивать результат выполнения автовекторизованного кода на одном ядре с кодом, векторизованным вручную на 60 ядрах, общий эффект от оптимизации - это ускорение всего расчета в 25,41 раз, но результат по времени существенно хуже, чем на процессоре Xeon E5, так как слишком медленно (в 20 раз медленнее) выполняются операции с тригонометрическими функциями в расчете поворотов. Также тестирование показало, что использование нескольких потоков на ядро не оправдано для данной задачи и не приводит к улучшению общего результата.

Наиболее интересным результатом является расчет на последней вычислительной архитектуре от Интел – KNL. Тестирование было проведено при максимальной загрузке ядер одним и несколькими потоками. Конфигурация быстрой памяти MCDRAM – адресуемая, конфигурация сетки ядер – «каждый с каждым». Результаты для KNL приведены в таблицах 6-7.

Таблица 6. Результаты выполнения расчета на одном ядре и на всех ядрах процессора Intel Xeon Phi KNL

все ядра/потоки	KNL auto, t, c	KNL FMA 64/64, t, c	KNL AVX512 64/64, t, c	KNL, ускорение
повороты	7,4	3,3	3,3	2,24
правые части	131,9	1,47	1,43	92,24
интегрирование	3,77	0,06	0,08	47,13
Всего	143,07	4,83	4,81	29,74

KNL поддерживает предыдущие векторные инструкции и полностью совместим с x86, в отличие от KNC. Поэтому сравнение было проведено для распараллеленного векторизованного кода с помощью FMA и с помощью AVX512, так же как и при сравнении векторизованных версий. В отличие от первого случая при распараллеливании компилятор при использовании FMA оптимизирует код так же, как и при использовании AVX512, следовательно, в сложных алгоритмах упаковки строк нескольких матриц в один регистр смысла нет, что видно из табли-

цы 6. Параллельный код при этом выполняется всего в 1.5 раза медленнее, чем на Intel Xeon E5, при этом ускорение по сравнению с автовекторизованным кодом на одном ядре составляет 30 раз. Использование дополнительных потоков на ядро приводит, в отличие от KNC к существенному снижению производительности, поэтому также не имеет смысла для данной задачи, что видно в таблице 7.

Таблица 7. Результаты выполнения расчета на одном ядре и на всех ядрах процессора Intel Xeon Phi KNL (с гипертредингом)

все ядра/потоки	KNL auto, t, c	KNL FMA 64/128, t, c	KNL, ускорение (64/128)
повороты	7,4	4,84	1,53
правые части	131,9	2,34	56,37
интегрирование	3,77	0,08	47,13
Всего	143,07	7,26	19,71

Итоговые результаты сравнения трех архитектур приведены в таблице 8, из которой видно, что архитектура KNL существенно производительнее, чем KNC, в процедуре расчета правых частей даже опережает Intel Xeon E5, но в целом обе архитектуры медленнее Xeon E5 для рассматриваемой задачи. KNC медленнее в 8 раз, KNL – в 1.5 раза.

Таблица 8. Соотношение времени выполнения наилучших версий (во сколько раз Xeon Phi медленнее Xeon E5)

	KNC / Xeon E5	KNL / Xeon E5
повороты	16,23	2,97
правые части	3,64	0,72
интегрирование	2,17	1,33
всего	8,02	1,52

6. Выводы и перспективы дальнейших исследований

Векторизация кода решателя показала, что на KNL задача решается существенно быстрее, чем на KNC. Ядро KNL уступает по скорости ядру Xeon E5 всего в 3.5 раза. Весь потенциал процессора можно раскрыть только при использовании распараллеливания на все физические ядра.

Ручная векторизация дала ускорение вычислений в решателе в 2-3 раза по сравнению с автовекторизацией современным компилятором Интел (Intel Parallel Composer 2017).

В перспективе можно рассчитывать, что параллельная версия кода рассматриваемого решателя (в результате дальнейшей оптимизации алгоритма) будет работать на KNL сопоставимо с 10-ядерным CPU Intel Xeon E5.

Дальнейшая оптимизация может быть связана с изменением алгоритма распараллеливания с использованием декомпозиции для снижения количества синхронизаций в параллельном коде. Имеет также смысл оптимизировать расчет и применение матриц поворота. Это может дать хороший результат при большом числе потоков и возможно приблизить скорость вычисления на Phi (на KNL, отчасти на KNC) к скорости вычисления на Intel Xeon E5.
Работа выполнена при финансовой поддержке РФФИ - проекты №№ 16-47-340385, 16-07-00534, 15-01-04577, 15-07-06254 – и администрации Волгоградской области – проект № 16-47-340385.

Литература

1. Гетманский В.В., Горобцов А.С., Измайлов Т.Д. Распараллеливание расчёта напряжённо-деформированного состояния тела в многотельной модели методом декомпозиции расчётной области // Известия ВолгГТУ. Серия "Актуальные проблемы управления, вычислительной техники и информатики в технических системах". Вып. 16, ВолгГТУ, № 8 (111), 2013, с. 5-10.
2. Горобцов А.С., Гетманский В.В., Андреев А.Е., Doan D.T. Simulation and Visualization Software for Vehicle Dynamics Analysis Using Multibody System Approach // Creativity in Intelligent Technologies and Data Science. CIT&DS 2015: Proceedings / ed. by A. Kravets et. al., Springer International Publishing, Switzerland, 2015, p. 379-391.
3. Куликов И.М., Черных И.Г., Ненашев В.Е., Катышева Е.В. Математическое моделирование столкновения галактик на гибридных суперэвм с ускорителями Intel Xeon Phi // Суперкомпьютерные дни в России / Суперкомпьютерный консорциум университетов России, Федеральное агентство науч. организаций России, МГУ им. М.В. Ломоносова. - Москва, 2015. - С. 226-237.
4. Киреев С.Е. Оптимизация для кластера с ускорителями Xeon Phi задачи фильтрации водно-нефтяной смеси через эластичную пористую среду // Вычислительные методы и программирование : новые вычислительные технологии, т. 16, НИВЦ МГУ им. М.В. Ломоносова, № 2, 2015, с. 177-186.
5. Харченко С.А., Ющенко А.А. Параллельная реализация алгоритма разреженного QR разложения для прямоугольных верхних квази-треугольных матриц со структурой типа вложенных сечений // Суперкомпьютерные дни в России / Суперкомпьютерный консорциум университетов России, Федеральное агентство науч. организаций России, МГУ им. М.В. Ломоносова. - Москва, 2015. - С. 316-323.
6. Гетманский В.В., Андреев А.Е, Мовчан Е.О. Особенности применения различных наборов векторных инструкций для оптимизации кода расчёта динамики систем тел // Суперкомпьютерные дни в России = Russian Supercomputing Days : тр. междунар. конф. (г. Москва, 26-27 сентября 2016 г.) / Суперкомпьютерный консорциум университетов России, Федеральное агентство науч. организаций России, МГУ им. М.В. Ломоносова. - Москва, 2016. - С. 365-372.
7. Андреев А.Е., Гетманский В.В., Насонов А.А., Мовчан Е.О., Харьков Е.С. Векторизация алгоритмов анализа динамического напряженно-деформированного состояния с использованием широких векторных регистров. // Параллельные вычислительные технологии – XI международная конференция, ПаВТ'2017, г. Казань, 3–7 апреля 2017 г. Короткие статьи и описания плакатов. - Челябинск: Издательский центр ЮУрГУ, 2017. – С. 243-254.
8. Гетманский, В.В. Ускорение расчёта динамического напряжённо-деформированного состояния с помощью наборов векторных инструкций / В.В. Гетманский, Е.О. Мовчан, А.Е. Андреев // Известия ЮФУ. Технические науки. - 2016. - № 11 (184). - С. 27-39.

Vectorization of dynamic stress-strain simulation algorithms on x86 architectures with SIMD instructions support

V.V. Getmanskiy, A.E. Andreev, E.S.Kharkov, E.O. Movchan

Volgograd State Technical University

When modeling the complex technical objects represented by a multibody systems, in particular, in the analysis of the dynamic stress-strain state of such objects, it is required to perform many matrix and vector operations with matrices of small dimension at each step of integration of systems of differential equations. This paper describes the reduction of execution time of these operations on the x86 processor architectures, using SIMD instruction sets SSE, AVX, AVX2, AVX512 for the Intel® Xeon ® central processors, coprocessors Intel® Xeon Phi™ (KNC) and 2nd generation Intel® Xeon Phi™ CPU processors (KNL).

Keywords: code optimization, vectorization, vector registers, dynamic stress-strain state, AVX-512, KNC, KNL

References

1. Getmanskiy V.V., Gorobcov A.S., Izmaylov T.D. Rasparallelivanie rascheta napryazhenno-deformirovannogo sostoyaniya tela v mnogotel'noy modeli metodom dekompozicii raschetnoy oblasti // Izvestiya VolgGTU. Seriya "Aktual'nye problemy upravleniya, vychislitel'noy tekhniki i informatiki v tekhnicheskih sistemah". Vyp. 16, VolgGTU, № 8 (111), 2013, pp. 5-10.
2. Gorobcov A.S., Getmanskiy V.V., Andreev A.E., Doan D.T. Simulation and Visualization Software for Vehicle Dynamics Analysis Using Multibody System Approach // Creativity in Intelligent Technologies and Data Science. CIT&DS 2015: Proceedings / ed. by A. Kravets et. al., Springer International Publishing, Switzerland, 2015, p. 379-391.
3. Kulikov I.M., Chernyh I.G., Nenashev V.E., Katysheva E.V. Matematicheskoe modelirovanie stolknoveniya galaktik na gibridnyh superehvm s uskoritelyami Intel Xeon Phi // Superkomp'yuternye dni v Rossii / Superkomp'yuternyj konsorcium universitetov Rossii, Fede-ral'noe agentstvo nauch. organizacij Rossii, MGU im. M.V. Lomonosova. - Moskva, 2015. - pp. 226-237.
4. Kireev S.E. Optimizaciya dlya klastera s uskoritelyami Xeon Phi zadachi fil'tracii vodno-neftyanoj smesi cherez ehlastichnuyu poristuyu sredu // Vychislitel'nye metody i programmirovanie : novye vychislitel'nye tekhnologii, t. 16, NIVC MGU im. M.V. Lomonosova, № 2, 2015, pp. 177-186.
5. Harchenko S.A., Yushchenko A.A. Parallel'naya realizaciya algoritma razrezhennogo QR razlozheniya dlya pryamougol'nyh verhnih kvazi-treugol'nyh matric so strukturoj tipa vlozhennyh sechenij // Superkomp'yuternye dni v Rossii / Superkomp'yuternyj konsorcium universitetov Rossii, Federal'noe agentstvo nauch. organizacij Rossii, MGU im. M.V. Lomonosova. - Moskva, 2015. - C. 316-323.
6. Getmanskiy V.V., Andreev A.E, Movchan E.O. Osobennosti primeneniya razlichnyh naborov vektornykh instrukciy dlya optimizacii koda rascheta dinamiki sistem tel / // Superkomp'yuternye dni v Rossii = Russian Supercomputing Days : tr. mezhdunar. konf. (g. Moskva, 26-27 sentyabrya 2016 g.) / Superkomp'yuternyy konsorcium universitetov Rossii, Federal'noe agentstvo nauch. organizaciy Rossii, MGU im. M.V. Lomonosova. - Moskva, 2016. - pp. 365-372.

7. Andreev A.E, Getmanskiy V.V., Nasonov A.A., Movchan E.O., Kharkov E.S. Vectorizacia algorithmov analiza dinamicheskogo napryazhenno-deformirovannogo sostoyanya s ispolzovanyem schirokykh vektornykh registrov. // Parallelnye vytcheslytelnye teknnologii – XI mezdunar. konf., PaVT'2017, g. Kazan, 3–7 aprelya 2017 g. - Chelyabinsk: Izd. Centr YuRGU, 2017. – pp. 243-254.
8. Getmanskiy, V.V. Uskorenie rascheta dinamicheskogo napryazhenno-deformirovannogo sostoyaniya s pomoshch'yu naborov vektornyh instrukciy / V.V. Getmanskiy, E.O. Movchan, A.E. Andreev // Izvestiya YUFU. Tekhnicheskie nauki. - 2016. - № 11 (184). - S. 27-39.