# Особенности применения различных наборов векторных инструкций для оптимизации кода расчета динамики системы тел

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

Волгоградский государственный технический университет

Рассмотрена разработка эффективного кода для расчета динамики системы тел со связями. Разрабатываемый расчетный модуль предназначен для определения динамического напряженно-деформированного состояния тел в сложных механизмах. Используемый алгоритм имеет ограничения по масштабируемости и максимальному ускорению при его распараллеливании. Для дальнейшей оптимизации кода использованы наборы векторных инструкций AVX, SSE, FMA, KNC. Получены результаты по ускорению, проведено сравнение с автовекторизацией.

*Ключевые слова:* автовекторизация, оптимизация кода, интринсики, регистры, динамика системы тел.

## 1. Введение

Зачастую в процессе проектирования механической системы или механизма появляется необходимость оценить напряжения и деформации, возникающие в узле под действием внешних сил. Для решения этой задачи используется метод дискретных элементов [1].

Данный метод заключается в моделировании динамики системы тел со связями, приближенных дискретными элементами, расположенными в ячейках трехмерной регулярной ортогональной сетки. Каждый элемент представляет собой АТТ (абсолютно твердое тело) заданного размера и массы, имеет 6 степеней свободы и от 1 до 6 связей с соседними элементами.

Масштабируемость такого алгоритма ограничена – по результатам исследований по распараллеливанию расчета на системах с общей и распределенной памятью (с использованием технологий OpenMP и MPI) [1-4] было достигнуто максимальное ускорение в 10 раз на 24 узлах без сопроцессоров, при дальнейшем увеличении числа узлов увеличение ускорения не наблюдается.

Следующим шагом по ускорению расчета является оптимизация кода. В данной работе для этого используется векторизация [5] как актуальный и эффективный способ повышения производительности вычислений на современных процессорах. С развитием технологий, производительность ядер процессоров традиционной архитектуры повышается благодаря увеличению разрядности векторных регистров, поэтому направление настоящей работы является перспективным для следующего поколения процессоров.

## 2. Метод дискретных элементов

В данной работе рассматривается решатель, выполняющий 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 (АТТ) 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.

В ходе расчета НДС (напряженно-деформированного состояния) необходимо смоделировать совокупность внутренних напряжений и деформаций, возникающих в теле в результате действия на него некоторых внешних сил, то есть получить тензоры напряжений ($\mathbf{T}_\sigma$) и деформации ($\mathbf{T}_\varepsilon$) в некоторый момент времени.

## 2.1 Общее описание модели решения

В методе дискретных элементов для расчета динамического НДС в упругом теле оно представляется как совокупность дискретных элементов кубической формы со стороной, равной шагу пространственной дискретизации. Эти элементы формируют в расчетной области геометрию модели.

Матрица масс в такой постановке задачи является диагональной, что значительно упрощает расчет. Таким образом, движение каждого дискретного элемента описывается обыкновенным дифференциальным уравнением следующего вида:

$$\mathbf{M}\ddot{\mathbf{y}} = \mathbf{q}(\dot{\mathbf{y}}, \mathbf{y}, t) - \mathbf{M}\mathbf{a}(t) + \mathbf{s}(\dot{\mathbf{y}}, \mathbf{y}), \qquad (1)$$

где $\mathbf{y}$ – координаты дискретных элементов, $\mathbf{M}$ – матрица инерции, $\mathbf{a}$ – вектор ускорений твердого тела, входящего в полную динамическую модель, $\mathbf{s}(\dot{\mathbf{y}}, \mathbf{y})$ – силы стабилизации.

Функция $\mathbf{q}(\dot{\mathbf{y}}, \mathbf{y}, t)$ описывает силы, возникающие из-за связей между дискретными элементами. Случайные возмущения, передаваемые от динамической модели, могут вызывать смещение дискретных элементов, поэтому для их привязки к опорному телу введены силы стабилизации $\mathbf{s}(\dot{\mathbf{y}}, \mathbf{y})$. Уравнение (1) записано в системе координат опорного тела, поэтому правая часть (1) также содержит силы инерции $\mathbf{Ma}(t)$ тела из динамической модели.

Для граничных узлов в правые части уравнений добавляются силы реакций, возникающие в связях между элементами. Уравнение для граничного элемента с использованием равномерного распределения внешних сил по граничным узлам имеет вид:

$$\mathbf{M}_i\ddot{\mathbf{y}}_i = \mathbf{f}(t)/|F| + \mathbf{q}_i(\dot{\mathbf{y}}, \mathbf{y}, t) - \mathbf{M}_i\mathbf{a}_i(t) + \mathbf{s}_i(\dot{\mathbf{y}}, \mathbf{y}),\ i \in F, \qquad (2)$$

где $|F|$ - количество узлов, входящих в границу, к которой приложена сила $\mathbf{f}$.

Уравнение (1) интегрируется явным методом Рунге-Кутта 4 порядка. Таким образом, данные, полученные из динамической модели, позволяют определить силы инерции и силы, возникающие в связях.

## 2.2 Пересчет матриц поворота

В описании модели используется два вида систем координат – глобальная, описывающая всю модель в целом, и локальные, описывающие положение каждого отдельного тела, ориентируясь по центру масс тела и центральным осям инерции. Введем следующие обозначения: глобальную СК обозначим за $O_0$, локальную СК опорного тела – $O_1$, дискретного элемента – $O_2$. Взаимное расположение данных систем координат и тел в них представлено на рис. 1.

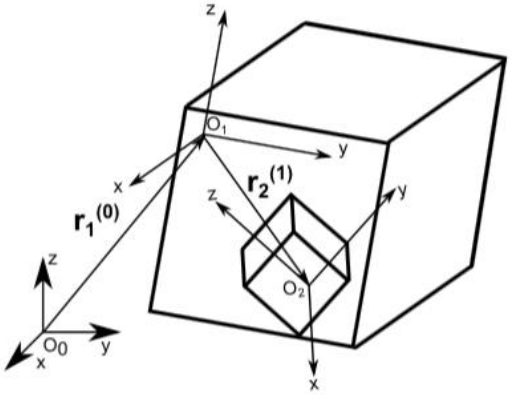



**Рис. 1.** Взаиморасположение глобальной и локальных систем координат

Так как все данные для расчета хранятся в глобальной системе координат, необходимо ввести преобразование положения тела из локальной СК в глобальную. Для этого используются матрицы поворота – квадратные матрицы, с помощью которых можно определить положение тела, описанного в одной системе координат, в другой СК.

Повороты выполняются на основе углов Эйлера (ψ, θ, и φ), позволяющих однозначно определить положение тела в данной системе координат. Новые значения углов можно получить, численно проинтегрировав кинематические уравнения Эйлера, определяющие проекции угловых скоростей тела с помощью углов Эйлера по следующим формулам:

$$\begin{cases} \dot{\psi} = \frac{\omega_1 \sin\varphi + \omega_2 \cos\varphi}{\sin\theta}, \\ \dot{\varphi} = \omega_3 - (\omega_1 \sin\varphi + \omega_2 \cos\varphi)\mathrm{ctg}\varphi, \\ \dot{\theta} = \omega_1 \cos\varphi - \omega_2 \sin\varphi \end{cases} \quad (3)$$

Эти операции необходимы для того, чтобы выполнить пересчет матриц поворота для каждого дискретного элемента. Данные матрицы имеют следующую форму:

$$A = \begin{bmatrix} \cos\theta\cos\psi & -\cos\theta\sin\psi & \sin\theta \\ \cos\varphi\sin\psi + \sin\varphi\sin\theta\cos\psi & \cos\varphi\cos\psi - \sin\varphi\sin\theta\sin\psi & -\sin\varphi\cos\theta \\ \sin\varphi\sin\psi - \sin\varphi\cos\theta\sin\psi & \sin\varphi\cos\psi + \cos\varphi\sin\theta\sin\psi & \cos\varphi\cos\theta \end{bmatrix} \quad (4)$$

### 2.3 Расчет деформаций и напряжений

Расчет правых частей уравнений (1) для каждого дискретного элемента осуществляется в несколько шагов. Сначала вычисляется матрица поворота для перевода дискретного элемента в систему координат опорного тела по формуле:

$$\mathbf{A}_{21} = \mathbf{A}_{02}^{\mathrm{T}} \mathbf{A}_{01}, \quad (5)$$

где первая цифра индекса означает номер СК, в которую необходимо перевести положение тела, вторая цифра – текущую СК.

С использованием параметров дискретных элементов и полученных по (4) матриц поворота рассчитываются деформации компонент:

$$\mathbf{d}_{\mathrm{sl}}^{(1)} = \mathbf{C}_1^{(1)} - \mathbf{A}_{21}^{\mathrm{T}}\mathbf{C}_2^{(2)} - \mathbf{A}_{01}^{\mathrm{T}}(\mathbf{P}_2 - \mathbf{P}_1)^{(0)}, \quad (6)$$

$$\mathbf{d}_{vl}^{(1)} = \mathbf{W}_1^{(1)} \times \mathbf{C}_1^{(1)} - \mathbf{A}_{21}^{\mathrm{T}}(\mathbf{W}_2^{(2)} \times \mathbf{C}_2^{(2)}) - \mathbf{A}_{01}^{\mathrm{T}}(\mathbf{V}_2 - \mathbf{V}_1)^{(0)}, \quad (7)$$

$$\mathbf{d}_{sa}^{(0)} = \mathbf{R}_1^{(0)} - \mathbf{R}_2^{(0)}, \quad (8)$$

$$\mathbf{d}_{va}^{(0)} = \mathbf{W}_1^{(0)} - \mathbf{W}_2^{(0)}, \quad (9)$$

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 СК, $\mathbf{W}_1^{(1)}$ и $\mathbf{W}_2^{(2)}$ – угловые скорости, $\mathbf{V}_1^{(0)}$ и $\mathbf{V}_2^{(0)}$ – линейные скорости дискретных элементов.

Выражение (6) задает линейный сдвиг дискретного элемента, (7) – изменение линейной скорости, (8) – угловой сдвиг и (9) – 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.

## 3. Векторизация решателя динамического НДС

Из формул (1) и (6-9) видно, что основные вычисления составляют матрично-векторные операции. В данном случае удобно применить векторизацию к расчету правых частей и численному интегрированию, так как эти операции выполняются независимо над разными наборами данных. Вычисления преобразований координат производятся с помощью матриц поворота (4), размерность которых 3x3, а не полных матриц преобразования в трехмерном пространстве размерностью 4x4 из-за особенностей вычисления неизвестных в механической системе. Одинаковой является процедура вычисления поступательных и вращательных степеней свободы

каждого дискретного элемента, но между собой они принципиально отличаются, кроме процедуры численного интегрирования. В процедуре численного интегрирования операции вычисления сумм и произведений производятся над полными векторами неизвестных, состоящих из всех степеней свободы дискретного элемента. При этом вычисленные вектора неизвестных скоростей и перемещений используются на следующей итерации для пространственных преобразований, поэтому целесообразно хранить вектора сгруппированными по координатам, то есть тройками $(x, y, z)$ для поступательных степеней свободы и $(rx, ry, rz)$ – для вращательных.

Векторные регистры процессора имеют разрядность, кратную степени двойки, поэтому, на первый взгляд можно было бы эффективно векторизовать операции с полной матрицей пространственных преобразований в однородных координатах, но ввиду отмеченных выше причин для всей процедуры вычислений и минимизации операций выделения и копирования памяти использован способ хранения векторов последовательно по координатам с пропусками в 1 элемент для кратности вектора разрядности используемого регистра.

### 3.1 Описание способа организации памяти

В данном методе важную роль играет организация памяти. Практически каждая операция расчета включает загрузку данных из памяти и выгрузку в нее результатов. Поэтому работа с памятью должна быть максимально эффективной.

По умолчанию используется не выровненное выделение памяти, при котором блоки данных не умещаются в границах машинных слов или сдвинуты относительно этих границ. Работа с невыровненными данными осуществляется существенно дольше, чем с выровненными, поэтому первым шагом для оптимизации кода является выровненное выделение памяти под массивы данных, для чего используются команды вида:

```
_varX = (double*)_aligned_malloc(varSize, 32);
```

С использованием векторных регистров связана также проблема размерности матриц, связанная с необходимостью загрузки в регистр на 2, 4 или 8 вещественных элементов двойной точности данных, сгруппированных по три элемента. Для эффективной загрузки в регистр размерность матриц и векторов должна быть 3x4 и 1x4 соответственно, чтобы загружать их по строкам.

Этого можно достичь, если дополнить каждую строку фиктивным четвертым элементом, равным 0, который фактически не участвует в операциях и не влияет на результат.

После этого строка матрицы либо целиком помещается в векторный регистр (для AVX), либо занимает два полных регистра меньшего размера (для SSE), можно также загружать в регистр две строки одновременно (для KNC). Таким образом, хранение векторов для координат *n* дискретных элементов в выровненной памяти имеет вид $(\boldsymbol{x_0, y_0, z_0, 0, rx_0, ry_0, rz_0, 0, x_1, y_1, z_1, 0, \dots, x_n, y_n, z_n, 0, rx_n, ry_n, rz_n})$ , а матриц поворота, соответственно: $(\boldsymbol{a_{00}, a_{01}, a_{02}, 0, a_{10}, a_{11}, a_{12}, 0, a_{20}, a_{21}, a_{22}, 0, \dots})$, где $\boldsymbol{a_{ij}}$ - элемент матрицы поворота дискретного элемента.

### 3.2 Анализ эффективности автовекторизации при оптимизации решателя

Современные компиляторы используют алгоритмы анализа кода, позволяющие проводить оптимизацию на этапе трансляции в машинные команды. Основная операция оптимизации, дающая существенное ускорение – это автовекторизация.

В таблице 1 представлены данные о времени выполнения расчета над одной и той же моделью на разных процессорах. Для сравнения использованы встроенный компилятор Microsoft C++ Compiler версии 18.0 (Visual Studio 2013) и компилятор фирмы Intel C++ версии 15.0 с ключами оптимизации по скорости (-O2) и полной оптимизации (-O3) и включенной поддержкой векторных команд.

**Таблица 1.** Результаты выполнения расчета с разными вариантами автовекторизации

| Процессор | Уровень оптимизации | Без автовекторизации | SSE | AVX | AVX2 |
|---|---|---|---|---|---|
| Intel Core i7 | Без оптимизации | 11.741 с | 11.834 с | 11.830 с | 11.466 с |

| | | | | | |
|---|---|---|---|---|---|
| CPU 3537U | По скорости (-O2) | 7.135 с | 7.060 с | 7.341 с | 7.074 с |
| | Полная (-O3) | 8.410 с | 7.555 с | 7.329 с | 7.288 с |
| Intel Xeon CPU E5-2650 v3 | Без оптимизации | 17.548 с | 17.217 с | 14.001 с | 12.561 с |
| | По скорости (-O2) | 12.825 с | 12.599 с | 9.261 с | 8.913 с |
| | Полная (-O3) | 12.729 с | 12.472 с | 9.199 с | 8.876 с |

Таким образом, максимальное ускорение, полученное с помощью автовекторизации, для данного решателя составляет около 1,3 раза при компиляции с полной оптимизацией и поддержкой расширения набора команд AVX2.

### 3.3 Описание векторизованного алгоритма численного интегрирования

Алгоритм численного интегрирования с помощью явного метода Рунге-Кутты 4-го порядка точности предполагает вычисление следующего приближения в 4 шага. На каждом шаге вычисляются коэффициенты, которые используются на следующих шагах, что требует обновления данных в памяти после каждого этапа.

Рассмотрим векторизацию данного метода на примере одного из шагов вычислений. Невекторизованный фрагмент программы, рассчитывающей 3-й шаг, выглядит так:

```
for (int j = 0; j < _nVariables; j++) {
    _hDDX3[j] = _varDDX[j] * _timeStep;
    _varX[j] = _initX[j] + (_initDX[j] + _hDDX2[j] * 0.5) * _timeStep;
    _varDX[j] = _initDX[j] + _hDDX3[j];
```

Этот же фрагмент с помощью AVX-интринсиков можно записать так:

```
for (int j = 0; j < _nVariables; j += 4) {
    Xtmp = _mm256_load_pd(_initX + j);
    DXtmp = _mm256_load_pd(_initDX + j);
    DDXtmp = _mm256_load_pd(_varDDX + j);

    hDDX3 = _mm256_mul_pd(DDXtmp, timeStep);
    _mm256_store_pd(_hDDX3 + j, hDDX3);
    tmp = _mm256_fmadd_pd(hDDX2, constant, DXtmp);
    tmp = _mm256_fmadd_pd(tmp, timeStep, Xtmp);
    _mm256_store_pd(_varX + j, tmp);
    _mm256_store_pd(_varDX + j, _mm256_add_pd(DXtmp, hDDX3));
```

Для других регистров меняются только названия инструкций и шаг цикла, так, для KNC:

```
for (int j = 0; j < _nVariables; j += 8){

    Xtmp = _mm512_load_pd(_initX + j); //. . .
```

Здесь также использованы FMA-интринсики, выполняющие совмещенное сложение-умножение за одну машинную команду. В идеале ускорение данной функции должно быть равно размеру использованных векторных регистров – 8 раз для KNC, 4 раза для AVX и 2 для SSE. Но на деле из-за большого числа операций загрузки и выгрузки данных достигаемый эффект не так значителен – 2,1 и 1,7 раза, соответственно, на AVX и SSE с помощью FMA-команд.

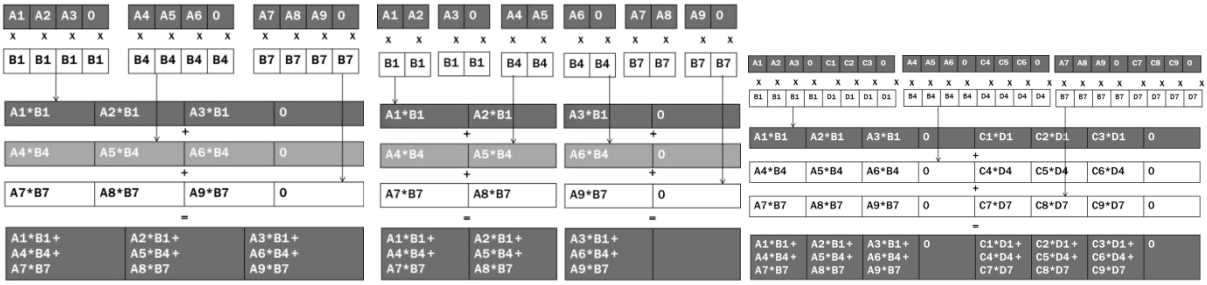
### 3.4 Описание векторизованного алгоритма вычисления правых частей

Наибольшие временные затраты при расчете НДС вызывает вычисление правых частей. Все операции в данной функции являются матричными или векторными по выровненной схеме, описанной выше.

Рассмотрим векторизацию матричных операций на примере транспонированного перемножения матриц:

```
MathHelpers::Mat3x4 matA21 = matA02.Tmul(matA01);
```

Для векторизации с помощью интринсиков применяется схема, представленная на рис. 2. На схеме приведен пример вычисления 1-й строки результирующей матрицы для SSE, AVX и KNC-инструкций.



а) б) в)

**Рис. 2.** Схема векторизованного перемножения матриц: а) с помощью регистров AVX, б) с помощью регистров SSE, в) с помощью регистров KNC

Для данной функции максимальное ускорение ограничивается размерностью матриц – 3 раза, и это ускорение обуславливает общее ускорение программы, так как операция является самой трудоемкой. Полученное ускорение за счет использования SSE-интринсиков – около 2,9 раз, AVX-интринсиков – примерно 3,3 раза. Для регистров KNC при условии одновременной обработки двух матриц/векторов ожидается ускорение порядка 4-4,5 раз.

## 4. Выводы и перспективы дальнейших исследований

Векторизация кода решателя дала следующие результаты:

- среднее ускорение в 1,3 раза с помощью автовекторизации;
- среднее ускорение в 2,4 раза на SSE;
- среднее ускорение в 2,8 раза на AVX;
- увеличение ускорения за счет FMA-инструкций до 3,0 раз.

Дальнейшая оптимизация алгоритмов для высокопроизводительных кластеров возможна за счет векторизации под новые архитектуры, например, поддерживающие AVX-512, и переноса кода пространственных преобразований для определения положения тел на графические вычислители и сопроцессоры.

Работа выполнена при финансовой поддержке РФФИ - проекты №№ 16-47-340385, 16-07-00534, 15-01-04577, 15-07-06254.

## 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. Getmanskiy V.V., Gorobtsov A.S., Sergeev E.S., Ismailov T.D., Shapovalov O.V. *Concurrent simulation of multibody systems coupled with stress-strain and heat transfer solvers*// Journal of Computational Science, 3(6) , 2012, p.492-497.
3. Сергеев, Е.С., Гетманский В.В., Горобцов А.С. *Перенос системы многотельной динамики на вычислительный кластер* // Научно-технические ведомости Санкт-Петербургского гос. политехн. ун-та, Вып. 101, 2010, с. 93-99.
4. 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.
5. Andreev, A., Nasonov, A., Novokschenov, A., Bochkarev, A., Kharkov, E., Zharikov, D., Kharchenko, S., Yuschenko, A. *Vectorization algorithms of block linear algebra operations using*

*SIMD instructions //* Communications in Computer and Information Science, 535, 2014, p. 323-340.

6. Zumbusch, G. *Vectorized Higher Order Finite Difference Kernels* // State-of-the-Art in Scientific and Parallel Computing (PARA), 2012, p. 343-357.

# Key features of multibody code vectorization using different instruction sets

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

Volgograd State Technical University

The efficient code development for multibody simulation is considered. The solver is developed for dynamic stress-strain simulation of bodies in complex mechanisms. Computational algorithm has limitations of scalability and maximal speedup in parallel implementation. Further optimization is performed using different sets of vector instructions such as SSE,AVX,FMA,KNC. Speedup results are achieved and compared with compiler auto-vectorization feature.

*Keywords:* auto-vectorization, code optimization, intrinsics, registers, multibody dynamics.

## References

1. Getmanskiy V.V., Gorobtsov A.S., Izmaylov T.D. *Rasparallelivanie raschyeta napryazhyenno-deformirovannogo sostoyaniya tela v mnogotel'noy modeli metodom dekompozitsii raschyetnoy oblasti* // Izvestiya VolgGTU. Ser. «Aktual'nye problemy upravleniya, vychislitel'noy tekhniki i informatiki v tekhnicheskikh sistemakh". Vyp. 16, VSTU, № 8 (111), 2013, p. 5-10.
2. Getmanskiy V.V., Gorobtsov A.S., Sergeev E.S., Ismailov T.D., Shapovalov O.V. *Concurrent simulation of multibody systems coupled with stress-strain and heat transfer solvers*// Journal of Computational Science, 3(6) , 2012, p.492-497.
3. Sergeev, E.S., Getmanskiy V.V., Gorobtsov A.S. *Perenos sistemy mnogotel'noy dinamiki na vychislitel'nyy klaster* // Nauchno-tekhnicheskie vedomosti Sankt-Peterburgskogo gos. politekhn. un-ta, Vyp. 101, 2010, p. 93-99.
4. Gorobtsov 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.
5. Andreev, A., Nasonov, A., Novokschenov, A., Bochkarev, A., Kharkov, E., Zharikov, D., Kharchenko, S., Yuschenko, A. *Vectorization algorithms of block linear algebra operations using SIMD instructions //* Communications in Computer and Information Science, 535, 2014, p. 323-340.
6. Zumbusch, G. *Vectorized Higher Order Finite Difference Kernels* // State-of-the-Art in Scientific and Parallel Computing (PARA), 2012, pp. 343-357.