1. P. Bahl and V. Padmanabhan, “RADAR: an in-building RF-based user location andtracking system,” in Proceedings of IEEE INFOCOM, pp. 775–784, Tel Aviv, Israel,March 2000.

2. N. Swangmuang and P. Krishnamurthy, “Location fingerprint analyses toward efficientindoor positioning,” in Proceedings of Sixth Annual IEEE International Conference onPervasive Computing and Communications, pp. 100–109, Hong Kong, March 2008.

3. R. Hansen and B. Thomsen, “Efficient and accurate wlan positioning with weighted graphs,” in *Mobile Lightweight Wireless Systems*, 2009, pp. 372–386.

4. T. Roos, P. Myllymki, H. Tirri, P. Misikangas, and J. Sievnen, “Aprobabilistic approach to wlan user location estimation,” *InternationalJournal of Wireless Information Networks*, vol. 9, no. 3, pp. 155–164,2002.

5. M. Youssef and A. Agrawala, “The horuswlan location determinationsystem,” *Wireless Networks*, vol. 14, no. 3, pp. 357–374, 2008.

ПОВЫШЕНИЕ БЫСТРОДЕЙСТВИЯ МАТРИЧНЫХ ОПЕРАЦИЙ В ЗАДАЧАХ РАСЧЕТА ЗАЗЕМЛИТЕЛЕЙ

***Шишигин Дмитрий Сергеевич**, аспирант*

Вологодский государственный технический университет, Россия

Заземляющее устройство (ЗУ) состоит из заземлителя и всех подключенных к нему проводников в воздухе предназначено для растекания токов молнии, короткого замыкания, импульсных помех в землю. Расчет ЗУ – трудоемкая задача с численным решением. Быстродействие компьютерной программы – один из основных критериев, определяющих ее конкурентоспособность, косвенный признак эффективности заложенных в ней алгоритмов. В данной работе мы рассмотрим методы и средства повышения производительности вычислений в программе ZYM.

При численных расчетах ЗУ дробится на элементы (короткие стержни), число которых составляет тысячи, десятки тысяч для современных подстанций. Все элементы связаны между собой гальваническими (через землю) и электромагнитными связями. Каждый элемент моделируется П–четырехполюсником с RLC элементами, поэтому математической моделью ЗУ является цепная схема замещения, а параметры ЗУ представлены квадратными, полностью заполненными матрицами сопротивлений растеканию **R**, индуктивностей **M**, и емкостей **C** [1]. Расчет напряжений и токов в этой схеме производится методами теории электрических цепей. При найденных токах электромагнитное поле ЗУ рассчитывается методами теории антенн. Проблема в том, что эти расчеты, включающие операции умножения, обращения действительных и комплексных матриц, очень трудоемки. Таких операций может быть много. Действия с комплексными числами увеличивают время в 4 раза. Поэтому существующие программы расчета ЗУ вынуждены ограничивать размерность задач, либо проводить расчеты без учета взаимных параметров.

Алгоритмическая оптимизация кода – один из основных способов повышения производительности. Согласно [2] 50% времени счета связаны с

4% кода, поэтому именно эти «узкие» места следует найти для оптимизации. В нашей задаче это расчет коэффициентов матрицы и операции с матрицами. При N=10000 требуется вычислить $3 \cdot 10^8$ коэффициентов массивов **R**,**M**,**C**, поэтому следуя рекомендациям [2] можно повысить быстродействие.

Идеи по оптимизации кода матричных операций находим в тексте программ математического пакета *Alglib* (распространяется свободно). Сравним привычный (по учебникам математики) алгоритм (рис.1а) с алгоритмом, где вычисления индексов двумерных матриц заменены действиями с указателями (рис.1б). Результаты сравнения представлены в таблице 1. Размерность матриц – 2000. Расчеты проведены на ноутбуке: Windows 7 64-bit, Intel Core i7 4x2.2 ГГЦ, ОЗУ 8 Гб.

Таблица 1. Сравнение стандартного алгоритма и алгоритма Alglib

Компилятор	Стандартный метод	Метод с указателями
MS Visual Compiler	94.3 сек	15.8 сек
Intel C++ Compiler	117.9 сек	13.2 сек
Delphi XE3 Compiler	96.1 сек	15.6 сек

Использование эффективного алгоритма перемножения матриц позволило повысить быстродействие в 6-9 раз по сравнению со стандартным методом, что существенно.

```
for (int i = 0; i < N; i++)
for (int j = 0; j < N; j++)
for (int k = 0; k < N; k++)
    C[i][j] += A[i][k] * B[k][j];
```

Рис.1а. Стандартный код программы перемножения матриц

```
for (int i = 0; i < N; i++)
for (int j = 0; j < N; j++)
{
    pB = &B[j][0];
    pC = &C[i][0];
    for (int k = 0; k < N; k++)
    {
        *pC += A[i][j] * (*pB);
        pB++;
        pC++;
    }
}
```

Рис.1б. Оптимизированный код программы перемножения матриц (пакет Alglib)

Заметим, однако, что алгоритмическая оптимизация обычно идет в ущерб простоте и читабельности кода и может привести к отрицательным результатам. В [2] отмечено, что применение оптимизирующего компилятора для простого кода может дать больший эффект, чем «хитрая» оптимизация циклов, после которой компилятор бессилен ускорить вычисления.

Для решения задач линейной алгебры используются специализированные математические пакеты. К их числу относится коммерческая библиотека *Intel Math Kernel Library* (далее *Intel MKL*), которая предназначена для

математических расчетов с высокой производительностью. В состав *Intel MKL* входят программы для работы с матрицами, быстрое преобразование Фурье, векторная математическая и статистические библиотеки, функции расширенной точности, программы для решения дифференциальных уравнений и методы оптимизации. Пользователи популярных математических пакетов, таких как Mathcad и Matlab вряд ли знают, что высокому быстродействию матричных операций они обязаны *Intel MKL*.

Данный пакет может быть подключен к собственной программе как любая динамическая библиотека. Рекомендуемые языки программирования – FORTRAN, на котором и создан пакет *Intel MKL*, а также C, C++, для которых разработаны интерфейсы.

Библиотека *Intel MKL* является потоко-безопасной, поддерживает распараллеливание и оптимизировано под многоядерные системы. Используя *Intel MKL*, разработчик может повысить производительность своего приложения за счет многопоточности и низкоуровневой оптимизации алгоритмов. Функции библиотеки оптимизированы для работы на процессорах *Intel*, однако они конкурентоспособны и на процессорах других производителей, что делает их универсальными.

Программирование с использованием *Intel MKL* имеет специфику. Мы уже отмечали снижение производительности из-за индексирования элементов больших двумерных массивов (рис.1). В *Intel MKL* используются только одномерные массивы, имеющие непрерывное расположение в оперативной памяти. Поэтому привычные двумерные массивы перед обращением к функциям *Intel MKL* нужно преобразовать в одномерные массивы, что требует дополнительной памяти, либо сразу записывать матрицы в одномерные массивы. Следует помнить, что, следуя правилам языка Фортран, двумерные массивы заполняются по столбцам.

Сопоставим быстродействие *Intel MKL* с аналогичным, но свободно распространяемым пакетом *Alglib* для основных матричных операций (Таблица 2). Размерность матриц – 2000. Расчеты проведены на ноутбуке: Windows 7 64-bit, Intel Core i7 4x2.2 ГГЦ, ОЗУ 8 Гб.

Таблица 2. Сравнение пакетов *Alglib* и *Intel MKL*

Операция	**AlgLib**	**Intel MKL**	**Сравнение**
Умножение матриц [**A**].[**A**]	11.8 сек	0.6 сек	19 раз
Решение СЛАУ [**A**].[**X**]=[**B**]	3.9 сек	0.3 сек	13 раз
Обращение матрицы $[\mathbf{A}]^{-1}$	28.8 сек	1.1 сек	26 раз
Решение СЛАУ [**C**].[**X**]=[**B**] с комплексными матрицами	46 сек	0.9 сек	51 раз
Обращение комплексной матрицы $[\mathbf{C}]^{-1}$	125 сек	3.6 сек	34 раз

Таким образом, применение *Intel MKL* существенно повышает быстродействие матричных вычислений по сравнению с пакетом *Alglib* и аналогичных. При разработке коммерческих программ с матричными операциями он, несомненно, стоит затраченных средств. Дополнительным доводом к его применению являются «рекомендации» Mathcad и Matlab.

В качестве примера рассчитаем сопротивление заземлителя (квадратная сетка со стороной 100 м) при уменьшении размера ячейки, что приводит к увеличению числа элементарных стержней N. Нам необходимо рассчитать коэффициенты матрицы собственных и взаимных коэффициентов **R**, число которых равно N^2, а затем решить СЛАУ для нахождения токов элементов и сопротивления заземлителя. С увеличением N время счета возрастает (рис.2), но существенно медленнее, чем квадратная парабола при расчете коэффициентов и кубическая парабола при решении СЛАУ, что объясняется многопоточностью вычислений и рассмотренными характеристиками *Intel MKL*.

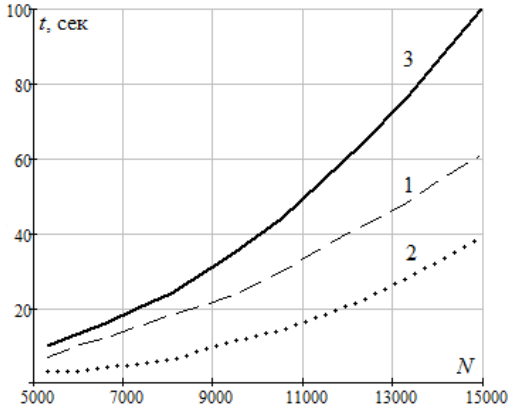

Рисунок 2. Время расчета коэффициентов матрицы размерности N (1), решения системы уравнений (2), суммарное (3)

В настоящее время существует тенденция к переходу от вычислений на центральных процессорах (CPU) к вычислениям на графических процессорах (GPU). Вычисления на GPU эффективны, когда задача может быть распараллелена на тысячи потоков и теоретически дает ускорение до 100 раз по сравнению с CPU (например, в областях вычислительной биологии и химии, моделирования динамики жидкостей, сейсмического анализа, трассировки лучей и др.). Сравним быстродействие умножения матриц тремя способами: алгоритмическая оптимизация (Alglib), специализированная библиотека (Intel MKL) и вычисления на GPU (Таблица 3). Размерность матриц – 3000. Расчеты проведены на ноутбуке: Windows 7 64-bit, Intel Core i7 4x2.2 ГГЦ, ОЗУ 8 Гб. Расчеты на GPU проводились на видеокарте: GeForce GTX 650 1Гб.

Таблица 3. Сравнение пакетов *Alglib* и *Intel MKL*

Операция	Alglib	Intel MKL	NVIDIA CUDA
Умножение матриц [**A**]·[**A**]	45.3 сек	2.5 сек	1.3 сек

Использование NVIDIA CUDA для вычислений на GPU позволило ускорить расчеты еще в 2 раза. Следует отметить, что эффективность вычислений на GPU зависит от многих факторов, таких как, производительность графической карты, размерность задачи, распараллеливаемость вычислений, т.д. Считается, что использование графических процессоров для вычислений в задачах линейной алгебры позволяет достичь ускорения в 3-15 раз по сравнению с вычислениями на CPU.

Вывод. Быстродействие компьютерной программы – один из основных критериев, определяющих ее конкурентоспособность, косвенный признак эффективности заложенных в ней алгоритмов. Существует резервы повышения быстродействия программы за счет оптимизации кода, распараллеливания алгоритмов, многопоточности вычислений. В задачах с матричными операциями большой размерности эффективными средствами повышения производительности расчетов является применение математической библиотеки *Intel Math Kernel Library*, а также использование *NVIDIA CUDA* для расчетов на графическом процессоре.

Список литературы

1. Шишигин, С.Л. Расчет заземлителей: учебное пособие / С.Л. Шишигин. – Вологда: ВоГТУ, 2012. – 121 с.

2. Макконнелл, С. Совершенный код : практическое руководство по разработке программного обеспечения / С. Макконнелл; [пер. с англ.] .— М.: Русская редакция, 2014.— 896 с.

ТЕХНОЛОГИЧЕСКИЕ ВОЗМОЖНОСТИ ПОВЫШЕНИЯ КАЧЕСТВА ОБРАБОТКИ ДЕТАЛЕЙ СВОБОДНЫМ АБРАЗИВОМ

***Шкурупий Валентин Григорьевич**, к.т.н.,*
доцент кафедры "Техника и технологии"
Харьковский национальный экономический университет, Украина

Шероховатость поверхностей деталей и физико-химическое состояние их поверхностного слоя обеспечивается на финишных операциях обработки. В настоящее время существует большое количество схем обработки свободным абразивом и реализованы соответствующие технологические системы [1 – 4]. Традиционное абразивное полирование – давно известный и широко применяемый способ обработки незакрепленными абразивными зернами. Известна эффективность обработки свободным абразивом при сглаживании исходной шероховатости поверхности. Однако этот процесс недостаточно управляем, что приводит к различным результатам.

Этому способствует отсутствие нормативов по выбору рабочих сред, полировальников, а также рекомендаций по достижению наименьших значений параметров шероховатости поверхностей.

Шероховатость поверхности после обработки будет определяться контактом обрабатываемой поверхности с формирующимся в динамике скоплением абразивных частиц. В перпендикулярной плоскости к обрабатываемой поверхности сечение абразивного скопления может быть представлено в виде элементарного мгновенного профиля абразивного конгломерата. Мгновенный профиль скопления абразивных частиц у обрабатываемой поверхности создается и фиксируется полировальником при традиционном абразивном полировании, видоизменяется по форме профиля сечения абразивного скопления, например, при наложении магнитного поля. Можно ожидать, что при наложении магнитного поля среднее арифметическое отклонение мгновенного профиля абразивного скопления будет увеличиваться и $R_{\max}$ то же, а это позволяет прогнозировать увеличение съема металла, то есть повышение производительности обработки. При наложении магнитного поля возможно уменьшение доли перекатывающихся зерен. Направленность перемещения абразивных зерен будет обеспечиваться напряженностью магнитного поля.

При вибрационной абразивной обработке направленное перемещение зерен будет определяться частотой инерционного вибратора и амплитудой перемещения рабочей камеры. Поэтому мгновенный профиль инструмента будет совершать перемещения в соответствии с вибрационными перемещениями, заданными исполнительными органами станка. В связи с тем, что рабочая камера колеблется в различных направлениях, воздействие абразивных частиц по поверхности усредняются, и мы получим характерную для этого способа обработки поверхность.

Для сглаживания поверхностного слоя при обработке свободным абразивом необходимо уменьшить шаржирование абразива и в полировальник, и в обрабатываемую поверхность. Этого можно достигнуть уменьшением давления полировальника на обрабатываемую поверхность, а также овализацией зерен абразивных порошков. Наличие укрупненной фракции в поставляемых промышленностью абразивных порошках ведет к царапанию обрабатываемых поверхностей и при длительном полировании происходит разрушение зерен, что приводит к уменьшению значений отношения $\frac{R_a}{R_{\max}}$, что также подтверждается увеличением количества царапин на полированной поверхности.

Проведенные нами ранее исследования позволили рекомендовать технологические среды на основе ультрадисперсных абразивов оксида алюминия, которые получают газодисперсным синтезом. Ультрадисперсным абразивам оксида алюминия свойственны: минимальная величина фракции