УДК 004.043:519.64+519.683:519.688

МИНИМИЗАЦИЯ ВРЕМЕНИ КОМПЬЮТЕРНОГО РЕШЕНИЯ ЗАДАЧИ ДИФРАКЦИИ НА ЭКРАНАХ МЕТОДОМ ОПТИМИЗАЦИИ ЛОКАЛЬНОСТИ ПАМЯТИ

Б. В. Паточкин
Харьковский национальный университет имени В. Н. Каразина
пл. Свободы, 4, г. Харьков, 61000, Украина. E-mail: equilibrium2702@gmail.com

Для компьютерных средств математического моделирования весьма актуальна задача экономии времени за счёт параллельных вычислений, ориентированных на архитектуру персональных компьютеров, имеющих 2–8 и более процессорных ядер. На основе анализа известных вычислительных методов, используемых при решении дифракционных задач на системе экранов методами дискретных особенностей, впервые разработана модификация таких вычислительных методов, основанная на локализации памяти при распараллеливании вычислений на разные ядра. Достигается комплексная экономия машинного времени на формирование матрицы линейной системы, её решение и численное построение диаграммы рассеяния. Эта модификация имеет форму параметризованной алгоритмической схемы и она прошла системную экспериментальную проверку (с уточнением параметров). В результате разработана параллельная модификация решения задачи дифракции E-поляризованной волны на контурах, в определённом смысле оптимальная для многоядерной архитектуры.

Ключевые слова: дифракция, МДО, время счёта, потоки, ядра, локальность памяти, тайлинг, кэш.

МІНІМІЗАЦІЯ ЧАСУ КОМПЬЮТЕРНОГО РІШЕННЯ ЗАДАЧІ ДИФРАКЦІЇ НА ЕКРАНАХ МЕТОДОМ ОПТИМІЗАЦІЇ ЛОКАЛЬНОСТІ ПАМ'ЯТІ

Б. В. Паточкін
Харківський національний університет імені В. Н. Каразіна
пл. Свободи, 4, м. Харків, 61000, Україна. E-mail: equilibrium2702@gmail.com

Для комп'ютерних засобів математичного моделювання досить актуальна задача економії часу за рахунок паралельних обчислень, орієнтованих на архітектуру персональних комп'ютерів, що мають 2–8 і більше процесорних ядер. На основі аналізу відомих обчислювальних методів, що використовуються при розв’язанні задач дифракції на системі екранів методами дискретних особливостей, вперше розроблена модифікація обчислювальних методів, яка заснована на локалізації пам'яті при розпаралелюванні обчислень на різні ядра. Досягається комплексна економія машинного часу на формування матриці лінійної системи, її розв’язку і чисельну побудову діаграми розсіювання. Ця модифікація має форму параметризованої алгоритмічної схеми і вона пройшла системну експериментальну перевірку (з уточненням параметрів). У результаті розроблена паралельна модифікація розв’язку задачі дифракції E-поляризованої хвилі на контурах, в певному сенсі оптимальна для багатоядерної архітектури.

Ключові слова: дифракція, МДО, час обчислення, потоки, ядра, локальність пам'яті, тайлінг, кеш.

АКТУАЛЬНОСТЬ РАБОТЫ. При всестороннем исследовании технических систем с применением компьютерных средств математического моделирования всё большую роль играют задачи экономии времени за счёт организации параллельных вычислений, ориентированных на архитектуру имеющихся машин. Широкое распространение получило использование четырех-, восьмиядерных (и более) компьютеров, чем повышается интерес к получению на них ускорения счёта пропорционального числу ядер, несмотря на проблему конкурентного доступа к общей памяти. При этом ни известные многочисленные результаты в стиле «неограниченного параллелизма» (что наглядно показано в [1]), ни результаты исследования возможностей ускорения решения разного типа задач на машинах специализированной архитектуры (как в [2]) не позволяют автоматически модифицировать всякий вычислительный метод под нужную архитектуру. Однако можно разрабатывать подходы, позволяющие проводить «параллельную» модификацию вычислительного метода на основе анализа подходящей параметризованной алгоритмической схемы. При этом окончательное уточнение параметров проводится с помощью вычислительных экспериментов на интересующей конечных пользователей платформе [3].

Такая концепция применяется нами к вычислительным методам дискретных особенностей (МДО) в задачах электродинамики [4]. В данной работе мы ограничимся примером варианта, в котором используются квадратуры интерполяционного типа для интегральных операторов с логарифмическим ядром [5].

В отличие от распределения параллельных вычислений на разные компьютеры (или узлы вычислительного кластера), использование нескольких процессорных ядер в составе одного компьютера (узла кластера) сталкивается с возможностью конфликта по разделяемым ресурсам, прежде всего по памяти. Не гарантируется, что время выполнения одной и той же программы с помощью многоядерного процессора всегда будет меньше, чем с помощью одноядерного процессора такой же тактовой частоты. С другой стороны, чем дольше процессор работает с кэшем без обращений к оперативной памяти, не нуждаясь в его обновлении, тем, при прочих равных условиях, быстрее идут вычисления. Это – так называемый эффект локализации памяти, и существует общая теория, как им следовало бы управлять [6]. Но даже при хорошей локализации разные ядра, в принципе, могут задерживать друг

друга, используя общий кэш. Из сказанного вытекает, что эффективные по скорости программные решения не могут не учитывать конкретные особенности вычислительной задачи и компьютеров, на которых она решается.

При реализации МДО особого внимания требуют, во-первых, алгоритм формирования линейных систем (СЛАУ), моделирующих сингулярные или гиперсингулярные интегральные и близкие им псевдодифференциальные операторы, и, во-вторых (в этой части мы стремились усилить прикладной результат [3]), – алгоритм решения СЛАУ.

Цель данной работы – модификация численного метода определения дискретных токов при решении дифракционных задач МДО, основанная на локализации памяти при распределении вычислений на разные ядра. Требуется обеспечить комплексную экономию машинного времени на формирование матрицы линейной системы, её решение и численное построения диаграммы рассеяния.

МАТЕРИАЛ И РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЙ.

1. Задача, численное решение которой нужно ускорить.

Пусть в плоской задаче дифракции электромагнитных волн на идеально проводящих контурах зависимость от времени задается множителем $e^{i\omega t}$. Амплитуды полного поля $\vec{E}^{\,tot} = \vec{E} + \vec{E}_0$, $\vec{H}^{\,tot} = \vec{H} + \vec{H}_0$, где $\vec{E}_0, \vec{H}_0$ – падающее поле, E, H - рассеянное, причем $E_0 = (0, 0, U_0)$, $H_0 = \left(\frac{1}{i\omega\mu}\frac{\partial U_0}{\partial y}, \frac{1}{i\omega\mu}\frac{\partial U_0}{\partial x}, 0 \right)$. Можно, как хорошо известно, считать, что $E^{tot} = (0, 0, U^{tot})$, $U^{tot}(X) = U(X) + U_0(X)$, $X \in R^2$.

Тогда рассеянное поле U удовлетворяет уравнению Гельмгольца всюду вне множества $C = \bigcup_{n=0}^{N-1} C_n$, где C_n – n-контур, а N – количество контуров:

$$\Delta U(X) + k^2 U(X) = 0, \quad x \in R^2 / \overline{C}, X = (x, y) \ . \tag{1}$$

На контуре C выполняется условие Дирихле

$$U(X)\big|_c = -U_0(X)\big|_c \tag{2}$$

на бесконечности – условия излучения Зоммерфельда

$$U(X) \underset{|X|\to\infty}{=} O\left(\frac{1}{\sqrt{r}}\right), r = |X| \cdot \tag{3}$$

$$\frac{\partial U(X)}{\partial r} - ikU(X) \underset{x\to\infty}{=} O\left(\frac{1}{r^{3/2}}\right), \tag{4}$$

а на всех концах контуров - условие на ребре

$$\int_\Omega \left(|U(X)|^2 + |\nabla U(X)|^2 \right) d\Omega < \infty, \tag{5}$$

где Ω это ограниченное открытое множество, содержащее конец контура.

2. Используемое интегральное уравнение с логарифмической особенностью.

Решение задачи (1)–(5) можно искать в виде следующего потенциала простого слоя [7]:

$$U(Y) = \int_C \mu(X) H_0^{(2)}\left(k|X - Y|\right) dl_X \ , \tag{6}$$

$\mu(X)$ – линейная плотность простого слоя в точке $X = (x_1, x_2) \in C$.

Формальная подстановка (6) в граничное условие (2) приводит к интегральному уравнению на системе контуров (ИУ):

$$\int_C \mu(X) H_0^{(2)}\left(k|X - Y|\right) dl_X = -U_0(Y), X, Y \in C \ . \tag{7}$$

Если ввести параметризацию контуров $C_n, n = 0, \ldots, N-1$,

$$C_n : \begin{cases} x_1 = x_1^{(n)}(t) \\ x_2 = x_2^{(n)}(t) \end{cases}, \tag{8}$$

то ИУ (7) становится эквивалентно системе интегральных уравнений на отрезках:

$$\sum_{n=0}^{N-1} \int_{-1}^{1} \mu^{(n)}(t) \mathrm{K}^{(n,m)}(t, \tau) L^{(n)}(t) dt = -U_0\left(x_1^{(m)}(\tau), x_2^{(m)}(\tau)\right), \ m = 0, \ldots, N-1, \quad \mu^{(n)}(t) = \mu\left(x_1^{(n)}(t), x_2^{(n)}(t)\right), \tag{9}$$

где $\mathrm{K}^{(n,m)}(t, \tau) = H_0^{(2)}\left(k \sqrt{\sum_{p=1}^{2} \left(x_p^{(n)}(t) - x_p^{(m)}(\tau) \right)^2} \right)$,

$$L^{(n)}(t) = \sqrt{\left(x_1^{(n)}(t)' \right)^2 - \left(x_2^{(n)}(t)' \right)^2} \ .$$

Как хорошо известно [8, 9],

$$H_0^{(2)}(z) \underset{z\to 0}{=} 1 - \frac{2i}{\pi}\left[\ln\frac{z}{2} + \gamma \right] + O\left(z^2 \ln z\right), \tag{10}$$

и ядра уравнений (9) имеют поведение в нуле

$$H_0^{(2)}\left(k \sqrt{\sum_{p=1}^{2} \left(x_p(t) - x_p(\tau) \right)^2} \right) \underset{\tau\to t}{=} a(t) + b(t) \ln|\tau - t| + O\left((\tau - t)^2\right) \ , \tag{11}$$

определяющее логарифмический характер особенности ядра:

$$a(t) = 1 - \frac{2i}{\pi}\left[\frac{1}{2} \ln\left(\sum_{p=1}^{2} \left(x_p'(t) \right)^2 \right) + \ln\frac{k}{2} + \gamma \right] \tag{14}$$

$$b(t) = -\frac{2i}{\pi} \cdot \tag{15}$$

Полагая

$$u^{(n)}(t) = \sqrt{1 - t^2}\, \mu^{(n)}(t) \cdot L^{(n)}(t), \tag{16}$$

где $\mu^{(n)}(t) \in \mathrm{L}_{2,\rho}, \rho(t) = \sqrt{1 - t^2}$ со скалярным произведением $(u, v)_\rho \equiv \int_{-1}^{1} u(t) \overline{v(t)} \rho(t) dt$,

можно придать системе (9) удобный вид:

$$\frac{-2i}{\pi}\int_{-1}^{1}\ln|\tau-t|\frac{u^{(m)}(t)}{\sqrt{1-t^2}}dt+ \\ +\sum_{n=0}^{N-1}\int_{-1}^{1}\frac{u^{(n)}(t)}{\sqrt{1-t^2}}K^{(m,n)}(\tau,t)dt=f^{(m)}(\tau)\ , \tag{17}$$

где $f^{(m)}(\tau)=-U_0\left(x_1^{(m)}(\tau),x_2^{(m)}(\tau)\right)$

$$K^{(m,n)}(\tau,t)=K_1^{(m,n)}(\tau,t),\ m\neq n\ , \\ K^{(n,n)}(\tau,t)=K_1^{(n,n)}(\tau,t)+\frac{2i}{\pi}\ln|\tau-t|\ , \tag{18}$$

где

$$K_1^{(m,n)}(\tau,t)=H_0^{(2)}\left(k\sqrt{\sum_{p=1}^{2}\left(x_p^{(n)}(t)-x_p^{(m)}(\tau)\right)^2}\right)\ .$$

3. Элементы матрицы СЛАУ задачи.

Перед получением дискретной системы (17), ее можно продифференцировать, желая работать с ядрами Коши. Однако из соображений оценить в дальнейшем оценку практической точности программных приложений за счёт диверсности [10], в данной работе используем (17) непосредственно.

Для дискретного моделирования СИУ и (17) используем известный интерполяционный подход [4, 5]. Введём приближения $u_P^{(n)}$ и f_P^n неизвестной функции $u^{(n)}, n=0..N-1$, и правой части $f^{(n)}, n=0..N-1$ их интерполяционными полиномами степени $P=P_n$, и аналогично для ядра $K^{(m,n)}(\tau,t)$, причём степени полиномов по переменным τ и t зависят от индексов n и m:

$$\frac{-2i}{\pi}\int_{-1}^{1}\ln|\tau-t|\frac{u_{P_{m-1}}^{(m)}(t)}{\sqrt{1-t^2}}dt+ \\ +\sum_{n=0}^{N-1}\int_{-1}^{1}\frac{u_{P_{n-1}}^{(n)}(t)}{\sqrt{1-t^2}}K_{P_{m-1},P_{n-1}}^{(m,n)}(\tau,t)dt=f_{P_{m-1}}^{(m)}(\tau)\ . \tag{19}$$

Используем квадратурную формулу интерполяционного типа для интеграла с логарифмическим ядром [5]:

$$\int_{-1}^{1}\ln|\tau-t|\frac{u_{n-1}(t)}{\sqrt{1-t^2}}dt= \\ =-\frac{\pi}{n}\sum_{k=0}^{n-1}u\left(t_k^n\right)\left[\ln 2+2\sum_{p=1}^{n-1}\frac{T_p(\tau)T_p\left(t_k^n\right)}{p}\right]\ , \tag{20}$$

где $u_n(t)$ – интерполяционный полином степени n функции $u(t)$, $T_p(x)$ – полином Чебышева и согласованную с этим методом квадратурную формулу интерполяционного типа для интегралов с гладкими функциями:

$$\int_{-1}^{1}\frac{u_{n-1}(t)}{\sqrt{1-t^2}}dt=\frac{\pi}{n}\sum_{k=0}^{n-1}u_{n-1}\left(t_k^n\right) \tag{21}$$

где $t_k^n=\cos\left(\pi\frac{2k+1}{2n}\right)$.

Применение в точках $t_p^{P_m}, p=0..P_m-1$, $m=0..N-1$ квадратурных формул (20-21), приводит к СЛАУ вида:

$$AX=B, \tag{22}$$

$$A=\left\{A_{n,m}\right\}_{n,m=0}^{N-1}, A\in C^{m,m}, m=\sum_{n=0}^{N-1}P_n\ , \tag{23}$$

где

$$A_{n,m}(i,j)=\frac{\pi}{P_n}\left(K^{(m,n)}\left(t_j^{(P_m)},t_i^{(P_n)}\right)\right),\ m\neq n\ , \\ A_{n,n}(i,j)=\frac{\pi}{P_n}\left(K^{(m,n)}\left(t_j^{(P_m)},t_i^{(P_n)}\right)+\right. \\ \left.+\frac{2i}{\pi}\left[LN^{(m,n)}(\mathrm{i,\ j})\right]\right)$$

В свою очередь,

$$LN^{(m,n)}(\mathrm{i,\ j})=\ln 2+2\sum_{p=1}^{P_n-1}\frac{T_p\left(t_j^{(P_m)}\right)T_p\left(t_i^{(P_n)}\right)}{p},$$

$$B=\left\{B_n\right\}_{n=0}^{N-1}, \tag{24}$$

где

$$B_m=-\left\{f^{(m)}\left(t_p^{P_m}\right)\right\}_{p=0}^{P_m-1}$$

$$X=\left\{X_n\right\}_{n=0}^{N-1}, \tag{25}$$

где

$$X_n=\left\{u^{(n)}\left(t_p\right)\right\}_{p=0}^{P_n-1}.$$

4. Использование локализации памяти для процесса вычисления элементов матрицы.

При дискретизации системы интегральных уравнений, большая часть времени тратится на заполнение матрицы. Согласно (23) заполняемая матрица состоит из блоков $A_{n,m}$, элементы которых вычисляются, используя контуры C_n и C_m. Так как на C_n мы всегда используем одни и те же точки $C_n^i(i=0..P_n-1)$, то логично вычислить их заранее.

Каждый элемент матрицы вычисляется независимо. При этом используется относительно небольшое количество данных C_n^i.

Расчет элементов матрицы происходит в четыре вложенных циклах

```
for (int i = 0; i < N; i++)
  for (int j = 0; j < N; j++)
    for (int ii = 0; ii < c[i].size(); ii ++)
      for (int jj = 0; jj < c[j].size(); jj++)
```

где N – количество контуров, c[k].size() – количество точек на k-том контуре.

Первый цикл отвечает за расчет полос состоящих из блоков $A_{n,m}$, второй – за расчет значений самих

блоков $A_{n,m}$, третий – за расчет строк состоящих уже из элементов матрицы (блока) и четвертый – уже за расчет самих элементов. В большинстве реальных задачах количество контуров варьируется от двух до 10, поэтому распараллеливание внутри первого цикла нецелесообразно, так как количество полос может часто окажется меньше количества ядер в системе. Распараллеливание внутри четвертого цикла, также не имеет смысла рассматривать, так как если задание для каждого потока не будет определено заранее (что часто бывает), то появится достаточно много накладных расходов на динамическое распределение нагрузки между потоками. Распараллеливание модификации по столбцам внутри блока (перестановка третьего и четвертого цикла) мы не рассматриваем, т.к. это заведомо медленнее. Остается всего два вида распараллеливания:

1) по блокам $A_{n,m}$ (внутри второго цикла);

2) по строкам внутри блока (внутри третьего цикла).

Из всего перечисленного выше видно, что скорость вычисления будет зависеть от локальности памяти [6] для элементов C_n^i. Можно выдвинуть гипотезу о том, что при распараллеливании по строкам внутри блока мы должны получить лучшую скорость счета по сравнению с распараллеливанием по блокам. Для проверки гипотезы эксперименты проводились при N=2, N=5 и N=10, количество точек дискретизации для каждого контура было P_n = 100, P_n =300, P_n =800.

Обозначим потоки внутри блока K_B, а потоки по строкам внутри блока – K_S.

Таблица 1 – Результаты экспериментов N=2, 5, 10

P_n	K_B	K_S	t, c, N=2	t, c, N=5	t, c, N=10
100	1	1	0,089	0,252	0,595
100	1	4	0,058	0,138	0,306
100	4	1	0,053	0,148	0,272
100	4	4	0,051	0,124	0,268
300	1	1	2,166	5,662	11,77
300	1	4	0,983	2,651	5,164
300	4	1	1,142	2,935	5,395
300	4	4	0,906	2,354	5,175
800	1	1	38,166	98,016	204,88
800	1	4	16,612	41,895	86,71
800	4	1	21,567	52,665	91,558
800	4	4	16,435	41,786	87,44

Каждый эксперимент состоял из четырех тестов:

1. без распараллеливания;
2. распараллеливание по строкам внутри блока;
3. распараллеливание по блокам;
4. распараллеливание одновременно двумя способами.

Эксперименты проводились на процессоре core i5-2430M (2 cores with hyper threading), компилятор Visual Studio 2013.

Результаты экспериментов подтверждают нашу гипотезу о том, что распараллеливание по строкам внутри блоков быстрее распараллеливания по блокам, причем разница в скорости может превышать 20 %.

5. Развитие метода практической оптимизации взаимодействия кэша с основной памятью.

Следуя далее в целом подходу к распараллеливанию решения СЛАУ (компактная схема Гаусса) работ [3, 11], рассмотрим, однако, перестановки циклов для нижнетреугольной матрицы, которые прежде на определенных основаниях априори считались бесперспективными для оптимизации. Речь идет о перестановках в следующем фрагменте алгоритма:

```
for (int i = 0; i < size; i++) {
 //для верхнетреугольной матрицы
 for (int j = i; j < size; j++)
  for (int k = 0; k < i; k++)
    matrix[i][j] -= matrix [i][k] * matrix [k][j];
 //для нижнетреугольной матрицы
 for (int j = i + 1; j < size; j++) {
  MatrixElementType sum = 0;
  for (int k = 0; k < i; k++)
   sum += matrix [j][k] * matrix [k][i];
  matrix [j][i] = (matrix [j][i] - sum) / matrix[i][i];
 }
 }
```

Здесь вложенность циклов соответствует порядку индексов (i, j, k).

Перспективным преобразованием этого фрагмента для нижнетреугольной матрицы является такое, которое соответствует порядку циклов (j, k, i). Мы заметили это по прямой аналогии с найденным ранее в [3] оптимальным (в численных экспериментах) порядком (i, k, j) для верхнетреугольной матрицы.

В этом перспективном варианте необходимо преодолеть недостаток, связанный с увеличением числа арифметических операций. Этого удалось достичь благодаря дополнительному разделению нижнетреугольной матрицы на небольшие подматрицы (их размер подобран в численных экспериментах) и разбиению вычислений на два этапа. Первый этап полностью аналогичен нахождению верхнетреугольной матрицы, а следовательно оптимальным порядком циклов по индексам станет (j, k, i), в примере ниже (ij, ik, ii). Второй этап позволил сэкономить вычисления за счет переноса в него одной общей операции деления:

```
for (int i = 0; i < bn; i++){
    ….
 for (int j = i + 1; j < bn; j++) {
  for (int k = 0; k < i; k++) {
   for (int ij=lBorder(j), rbj=rBorder(j); ij<rbj; ij++)
   // первый этап
    for (int ik=lBorder(k), rbk=rBorder(k); ik<rbk; ik++)
     for (int ii = lBorder(i), rbi = rBorder(i); ii < rbi; ii++)
      luMatrix[ij][ii] -= luMatrix[ij][ik] * luMatrix[ik][ii];
  }
  for (int ii = lBorder(i), rbi = rBorder(i); ii < rbi; ii++){
  // второй этап
   for (int ik = lBorder(i); ik < ii; ik++)
    for (int ij = lBorder(j), rbj = rBorder(j); ij < rbj; ij++)
     luMatrix[ij][ii] -= luMatrix[ij][ik] * luMatrix[ik][ii];
    for (int ij = lBorder(j), rbj = rBorder(j); ij < rbj; ij++)
     luMatrix[ij][ii] /= luMatrix[ii][ii];
  }
  }
  }
```

где bn – количество подматриц в строке или столбце, индексы i, j, k отвечают за перебор подматриц, lBorder(p) и rBorder(p) левая и правая граница подматрицы.

Покажем оптимальность в численных экспериментах. Сравним обычную реализацию, реализацию с улучшенной локальностью памяти и параллельную версию (табл. 2). Реализация с улучшенной локальностью памяти по своей структуре практически полностью соответствует параллельной. Эксперименты проводились на процессоре core i5-2430M (2 cores with hyper threading), компилятор – Visual Studio 2013.

Таблица 2 – Результаты экспериментов решения СЛАУ для разных реализаций

Размерность матрицы	1000	1731	2996	5186	8978
Обычная реализация	1,325	10,39	68,2	395,8	2172
Реализация с улучшенной локальностью памяти	0,409	2,141	11,4	59,3	310,3
Параллельная реализация	0,215	1,123	5,88	30,8	161,3

Как видно из табл. 2, при переходе от обычной реализации к реализации с улучшенной локальностью памяти наблюдается прирост производительности в 3–7 раз (за счет обработки матрицы по подматрицам и перестановки циклов). С увеличением размерности матрицы, производительности программной реализации возрастает. Эффект распараллеливания вместе с использованием эффекта локальности, как и ожидалось, соответствует у нас числу ядер.

Эффект сохраняется при использовании различных компиляторов, тестирование проводилось на gcc4.7, Visual Studio 2012, Visual Studio 2013 и mingww-64 (gcc4.8.2 posix threading model).

6. Диаграмма направленности, вычисление которой подлежит ускорению.

Единственная компонента напряженности электрического поля выражается так (6), (8), (16):

$$U\left(x_1,x_2\right)=\sum_{n=0}^{N-1}\left[\int_{-1}^{1}K^{(n)}(t)\frac{u^{(n)}(t)}{\sqrt{1-t^2}}dt\right], \qquad (26)$$

где $K^{(n)}(t)=H_0^{(2)}\left(k\sqrt{\sum_{p=1}^{2}\left(x_p^{(n)}(t)-x_p\right)^2}\right)$.

Отсюда, с учётом (21), данная компонента может быть вычислена приближенно:

$$U_m\left(x_1,x_2\right)=\sum_{n=0}^{N-1}\left[\frac{\pi}{P_n}\sum_{p=0}^{P_n-1}K^{(n)}(t)\,X_{n,p}\right]. \qquad (27)$$

Задавшись направлением – полярным углом ц, получают, как известно, диаграмму направленности:

$$F(\varphi)=\lim_{r\to\infty}\frac{U}{e^{-ikr}r^{-1/2}}. \qquad (28)$$

С учётом асимптотического поведения цилиндрических функций на бесконечности имеем:

$$F(\phi)=\sqrt{\frac{2}{\pi k}}\sum_{n=0}^{N-1}\left[\int_{-1}^{1}\exp\left(ikR(t)\right)\frac{u^{(n)}(t)}{\sqrt{1-t^2}}dt\right]$$

Тогда диаграмма направленности электрического поля приближенно вычисляется так:

$$F(\varphi)=\sqrt{\frac{2}{\pi k}}\sum_{n=0}^{N-1}\left[\frac{\pi}{P_n}\sum_{p=0}^{P_n-1}\exp\left(ikR(t)\right)X_{n,p}\right].$$

7. Сравнение скорости счета распараллеленной реализации с обычной.

Определим стандартное минимальное численное решение (СМЧР) дифракционной задачи как нахождение «дискретных токов» из дискретной модели уравнения (19) (решение СЛАУ) и по нему – поля в дальней зоне, т.е. диаграммы направленности по напряженности поля с шагом 1°. СМЧР делает осмысленным сравнение счета на основе разных методов. Сравним время обычной программной реализации МДО с реализацией по методу работы. Временем построения диаграмм направленности можно было пренебречь. Обозначим реализации: S – однопоточная без оптимизации, M – наша параллельная.

Таблица 3 – Время получения СМЧР при N = 1

P_n	Заполнение матрицы		решение СЛАУ		время	
	S	M	S	M	S	M
100	0,206	0,038	0,001	0,001	0,211	0,043
200	1,53	0,238	0,014	0,009	1,562	0,255
400	10,16	1,861	0,119	0,061	12,14	1,938
800	103,7	13,44	1,092	0,24	104,9	13,72
1200	346,4	43,42	4,152	0,693	350,6	44,16
1600	810,7	108	10,67	1,482	821,4	109,6
2000	1559	199,1	23,78	3,508	1582	202,7

Таблица 4 – Время получения СМЧР при N = 2

P_n	Заполнение матрицы		решение СЛАУ		время	
	S	M	S	M	S	M
100	0,404	0,088	0,014	0,009	0,426	0,105
200	3,196	0,495	0,121	0,048	3,334	0,56
400	24,66	3,491	1,038	0,238	26,045	4,079
800	209,8	27,54	11,07	1,494	221,5	29,7
1200	696,2	88,14	42,85	4,828	739,14	93,07
1600	1634	205,3	113,2	11,29	1746,9	216,8
2000	3122	403,3	250,1	21,9	3372,6	425,3

Таблица 5 – Время получения СМЧР при N = 5

P_n	Заполнение матрицы		решение СЛАУ		время	
	S	M	S	M	S	M
100	1,132	0,175	0,236	0,078	1,389	0,274
200	8,144	1,141	2,249	0,404	10,43	1,588
400	67,18	8,489	25,11	2,856	92,38	11,432
800	521,1	67,54	256,5	21,88	777,8	89,591
1200	1739	216,1	1038	73,83	2778	290,16
1600	4064	509,4	2513	174,1	6577	683,85
2000	7927	1020	5032	340,8	12960	1361

Таблица 6 – Время получения СМЧР при N = 10

P_n	Заполнение матрицы		решение СЛАУ		время	
	S	M	S	M	S	M
100	2,47	0,351	2,925	0,429	5,436	0,821
200	17,75	2,381	24,31	2,92	42,13	5,39
400	134,7	17,5	259,7	22,2	394,6	39,86
800	1075	134,2	2648	177,8	3723	312,4
1200	3573	447,9	8982	596,9	12556	1045
1600	8285	1044	21197	1419	29483	2464
2000	16009	2101	41493	2722	57503	4824

Таблица 7 – Зависимость времени счета от количества контуров для параллельной реализации

P_n	N			
	1	2	5	10
100	0,043	0,105	0,274	0,821
200	0,255	0,56	1,588	5,39
400	1,938	4,079	11,432	39,86
800	13,72	29,7	89,591	312,4
1200	44,16	93,07	290,16	1045
1600	109,6	216,8	683,85	2464
2000	202,7	425,3	1361	4824

Тесты проводились на процессоре Intel Xeon Processor E5645 (12M Cache, 2.40 GHz, 5.86 GT/s Intel QPI hyper threading) – 6 ядер.

Из табл. 3–7 видно, что эффект «распараллеливания» – ускорение счёта - тем выше, чем больше размерность матрицы дискретной модели, и при числе контуров > 5 или при порядке дискретизации > 250 оно не меньше числа процессорных ядер.

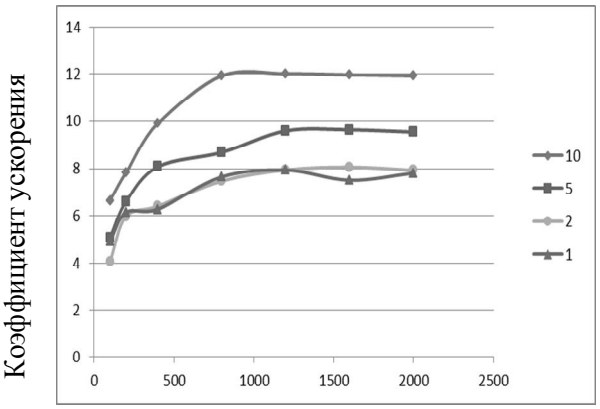

Рисунок 1 – Эффект от применения метода статьи (ускорение счёта) при 6 ядрах для разного числа контуров (1, 2, 5 и 10)

ВЫВОДЫ. Модифицирован под многоядерные компьютеры численный метод дискретных токов в задаче дифракции плоской электромагнитной волны на системе проводящих цилиндрических экранов. Время счета экономится за счет нового метода использования эффекта локальности памяти.

Разработанный в работе метод позволяет придавать существующим методам решения задач электродинамики на базе МДО, например, [4], форму вычислительных технологий исследования процессов излучения, рассеяния и прохождения волн в технических целях. Эти результаты могут быть развиты для 3D задач и других компьютерных архитектур.

По методу статьи разработана и практически оптимизирована по своим параметрам параллельная реализация вычислительного метода дискретных токов для задачи дифракции E-поляризованной волны на цилиндрических проводящих экранах с коэффициентом ускорения примерно равным или до двух раз превосходящим число ядер в системе.

Программа C++, реализующая этот метод, доступна [12] по лицензии GPL.

ЛИТЕРАТУРА

1. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. – С-Пб: БХВ-Петербург, 2002. – 608 с.

2. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем. – М.: Мир, 1991. – 367 с.

3. Оптимизация компактной схемы Гаусса для многоядерных процессоров / В.О. Мищенко, Б.В. Паточкин // Вестник ХНУ. – 2011. – № 981. – Вып. 18. – С. 70–81.

4. Математические вопросы метода дискретных токов. Обоснование численного метода дискретных особенностей решения двумерных задач дифракции электромагнитных волн / Ю.В. Гандель, С.В. Еременко, Т.С. Полянская: учеб. пособие. Ч. 2. – Харьков: ХГУ, 1992. – 145 с.

5. Гандель Ю.В. Введение в методы вычислений сингулярных и гиперсингулярных интегралов. – Харьков, 2001. – 92 с.

6. Лиходед Н.А. Методы распараллеливания гнезд циклов: курс лекций. – Минск: БГУ, 2008. – 100 с.

7. Уравнения математической физики / А.Н. Тихонов, А.А. Самарский. – М.: Наука, 1977. – 735 с.

8. Таблицы интегралов, сумм рядов и произведений. / Н.С. Градштейн, И.М. Рыжик. – М.: Физматгиз, 1962. – 1100 с.

9. Справочник по специальным функциям / М. Абрамовиц, И. Стиган. – М.: Наука, 1979. – 832 с.

10. Вычислительные эксперименты по точности моделирования МДО дифракции на идеально проводящих лентах / А.В. Боровинский, В.О. Мищенко, Б.В. Паточкин // Труды научно-технической конференции с международным участием «Компьютерное моделирование в наукоемких технологиях», г. Харьков, 28–31 мая 2014 г. – С. 45–48.

11. Паточкин Б.В. Согласование тайлинга с сокетным обменом на примере решения СЛАУ в вычислительном кластере // Труды научно-технической конференции с международным участием «Компьютерное моделирование в наукоемких технологиях» Харьков, 24–27 апреля 2012 г. – С. 334–337.

12. E-polarization task [Электронный ресурс]. – Режим доступа: https://github.com/Equilibrium2702/EPolarization.

MINIMIZING TIME OF COMPUTER SOLUTIONS OF DIFFRACTION PROBLEMS ON SCREEN BY THE LOCAL MEMORY OPTIMIZATION METHOD

B. Patochkin
V. N. Karazin Kharkiv National University
pl. Svobody, 4, Kharkiv, 61000, Ukraine. E-mail: equilibrium2702@gmail.com

Task of saving time via parallel computing on the modern computer architecture with 2-8 and more cores is very urgent for computer based mathematical modeling. Based on the analysis of the known computational methods used in the solution of diffraction problems on the system screens methods of discrete features (MDF), for the first time developed a modification of computational methods based on the memory localization with parallelization on different cores. Complex economy of computer time is achieved on the SLAE formation, its solution and the numerical construction of the scattering diagram. This modification is in the form of parameterized algorithmic scheme and it has passed systemic experimental verification (with specification of parameters). As a result, developed a parallel modification of the solution of the problem of diffraction of E-polarized wave on the contours, in some sense optimal for multi-core architecture. As a result, parallel modification of the solution of the E-polarized wave diffraction problem on the contours were developed, in some sense optimal for multi-core architecture.

Key words: diffraction, MDF, calculating time, threads, cores, memory locality, tiling, cache.

REFERENCES

1. Voevodin, V.V., Voevodin, Vl.V. (2002) *Parallel'nye vychislenija* [Parallel Computing], BHV-Petersburg, St. Petersburg, Russia.

2. Ortega, J. (1988) *Vvedenie v parallelnyie i vektornyie metodyi resheniya lineynyih sistem* [Introduction to Parallel and Vector Solution of Linear Systems], Plenum Press, New York.

3. Mishchenko, V.O., Patochkin, B.V. (2011) "Optimization of compact scheme Gauss for multicore processors", *Vestnik KhNU* , № 981, vol. 18, pp. 70–81.

4. Gandel, Y.V. Eremenko, S.V., Polyanska, T.S. (1992) *Matematicheskie voprosy metoda diskretnyh tokov. Obosnovanie chislennogo metoda diskretnyh osobennostej reshenija dvumernyh zadach difrakcii jelektromagnitnyh voln* [Mathematical problems of the discrete currents. Justification of the numerical method of discrete singularities solutions of two-dimensional problems of diffraction of electromagnetic waves], KhGU, Kharkiv, Proc. Allowance, Ukraine.

5. Gandel, Y.V. (2001) *Vvedenie v metody vychislenij singuljarnyh i gipersinguljarnyh integralov* [Introduction to computing singular and hypersingular integrals], Kharkiv, Ukraine.

6. Likhoded, N.A. (2008) *Metody rasparallelivanija gnezd ciklov: kurs lekcij* [Nests loop parallelizing Methods: lectures], BGU, Minsk, Belarus.

7. Tikhonov, A.N. Samarskiy, A.A. (1977) *Uravnenija matematicheskoj fiziki* [Equations of mathematical physics], Nauka, Moscow, USSR.

8. Gradshtein, N.S., Ryzhik, I.M. (1962) *Tablicy integralov, summ rjadov i proizvedenij* [Tables of integrals, sums of series and products], Fizmatgiz, Moscow, USSR.

9. Abramowitz, M., Steagan, I. (1979) *Spravochnik po spetsialnyim funktsiyamvedenie v parallelnyie i vektornyie metodyi resheniya lineynyih sistem* [Manual special functions], Nauka, Moscow, USSR.

10. Borovinskiy, A.V., Mishchenko, V.O., Patochkin, B.V. (2014) *Vychislitel'nye jeksperimenty po tochnosti modelirovanija MDO difrakcii na ideal'no provodjashhih lentah* [Computational experiments on the accuracy of diffraction modeling with method of discrete features on perfectly conducting tape]. *Trudy nauchno-tehnicheskoj konferencii s mezhdunarodnym uchastiem "Komp'juternoe modelirovanie v naukoemkih tehnologijah"*, Kharkiv, Ukraine, pp. 45–48.

11. Patochkin B.V. (2012) *Soglasovanie tajlinga s soketnym obmenom na primere reshenija SLAU v vychislitel'nom klastere* [Tailing and socket exchange coordination on SLAE solving example in computing cluster] *Trudy nauchno-tehnicheskoj konferencii s mezhdu-narodnym uchastiem "Komp'juternoe modeliro-vanie v naukoemkih tehnologijah"*, Kharkiv, Ukraine, pp. 334–337.

12. E-polarization task [Электронный ресурс]. – Mode access: https://github.com/Equilibrium2702/EPolarization.

Стаття надійшла 27.10.2014.