ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА»

М.А. ЧИЧЕВА

КОМПЬЮТЕРНАЯ АЛГЕБРА. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К КУРСОВОМУ ПРОЕКТИРОВАНИЮ

Утверждено Редакционно-издательским советом университета в качестве учебного пособия

САМАРА
Издательство СГАУ
2007

УДК 681.3
ББК 22.343
Ч-722

Инновационная образовательная программа "Развитие центра компетенции и подготовка специалистов мирового уровня в области аэрокосмических и геоинформационных технологий"

Рецензенты: д-р техн. наук А. Г. Храмов,
д-р техн. наук В. Г. Карташевский

Чичева М.А.

Ч-722 **Компьютерная алгебра. Методические указания к курсовому проектированию:** учеб. пособие / *М.А. Чичева.* – Самара: Изд-во Самар. гос. аэрокосм. ун-та, 2007. – 64 с. : ил. 11.

ISBN 978-5-7883-0633-9

Пособие представляет собой указания к курсовому проекту по дисциплине "Компьютерная алгебра", выполняемому в 9-м семестре студентами специальности "Прикладная математика и информатика" специализации "Математическое обеспечение систем обработки изображений". Пособие содержит весь необходимый теоретический и практический материал, включая сведения из теории алгебраических структур, быстрые алгоритмы двумерного дискретного преобразования Фурье, а также требования к программному обеспечению, созда-ваемому в ходе выполнения проекта, рекомендации по его тести-рованию и экспериментальному исследованию реализованного алгоритма. В приложения вынесены необходимые технические материалы.

Учебное пособие предназначено для студентов факультета инфор-матики, обучающихся по специальности "Прикладная математика и информатика".

УДК 681.3
ББК 22.343

ISBN 978-5-7883-0633-9

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Пособие представляет собой методические указания к курсовому проекту по дисциплине "Компьютерная алгебра", выполняемому в 9-м семестре студентами специальности "Прикладная математика и информатика" специализации "Математическое обеспечение систем обработки изображений".

Целью курсового проекта является программная реализация и исследование одного из алгоритмов двумерного дискретного преобразования Фурье (ДПФ) с представлением данных в одной из заданных алгебраических структур.

Пособие содержит весь необходимый теоретический и практический материал. Первая глава содержит сведения из теории алгебраических структур. В рамках пособия это вспомогательный материал, но он необходим для понимания остальных глав. Вторая глава посвящена схемам декомпозиции двумерного ДПФ, методике реализации соответствующих алгоритмов, способам учета вещественности входного сигнала. Приводится методика расчета вычислительной сложности быстрых алгоритмов. Третья глава содержит дополнительный материал, описывающий возможности параллельной реализации дискретного преобразования Фурье. Приведено исследование предполагаемой эффективности распараллеливания. В четвертой главе приводятся требования к программному обеспечению, создаваемому в ходе выполнения проекта, рекомендации по его тестированию и экспериментальному исследованию реализованного алгоритма.

В приложения вынесены необходимые технические материалы, такие как список тем для курсового проектирования, рекомендуемая дополнительная литература. Там же приведен образец оформления пояснительной записки и вспомогательные алгоритмы перестановок.

1 НЕОБХОДИМЫЕ СВЕДЕНИЯ ИЗ ТЕОРИИ АЛГЕБРАИЧЕСКИХ СТРУКТУР

Эта глава содержит необходимые для выполнения курсового проекта сведения об используемых алгебраических структурах: алгебре кватернионов и гиперкомплексной алгебре. Кроме того, в главу входит материал о представлении кватернионов в форме кодов Гамильтона-Эйзенштейна, необходимый при реализации преобразований по основаниям 3 и 6 (см. главу 2), а также о представлении гиперкомплексных чисел в прямой сумме комплексных алгебр, которое позволяет строить параллельные алгоритмы двумерного ДПФ (см. главу 3).

1.1 Алгебра кватернионов

Под телом H гамильтоновых кватернионов понимается четырехмерная ассоциативная алгебра над P

$$\mathrm{H} = \left\{ q = a + bi + cj + dk; a, b, c, d \in \mathrm{P} \right\}$$

с определяющими соотношениями для умножений базисных элементов $\{1, i, j, k\}$:

$$i^2 = j^2 = k^2 = -1, ij = -ji = k. \tag{1.1}$$

Поле комплексных чисел X канонически вкладывается в H:

$$a + bi \to a + bi + 0 \cdot j + 0 \cdot k. \tag{1.2}$$

Кроме того, справедливо соотношение

$$q = a + bi + cj + dk = (a + bi) + (c + di) j. \tag{1.3}$$

Операция сложения кватернионов осуществляется покомпонентно, а умножения - с учетом правил (1.1) и с приведением подобных членов.

Далее, отображения

$$\varepsilon_i : q \mapsto i^{-1} q i, \varepsilon_j : q \mapsto j^{-1} q j, \varepsilon_k : q \mapsto k^{-1} q k, \varepsilon_O : q \mapsto q \tag{1.4}$$

являются автоморфизмами H над P, причем

$$\begin{cases} \varepsilon_o(q) = a + bi + cj + dk, \\ \varepsilon_i(q) = a + bi - cj - dk, \\ \varepsilon_j(q) = a - bi + cj - dk, \\ \varepsilon_k(q) = a - bi - cj + dk. \end{cases} \tag{1.5}$$

Система уравнений (1.5), рассматриваемая относительно a, b, c, d, разрешима при любых значениях левых частей и требует для решения не более четырех вещественных умножений:

$$\begin{cases} 4a = \varepsilon_o(q) + \varepsilon_i(q) + \varepsilon_j(q) + \varepsilon_k(q), \\ 4bi = \varepsilon_o(q) + \varepsilon_i(q) - \varepsilon_j(q) - \varepsilon_k(q), \\ 4cj = \varepsilon_o(q) - \varepsilon_i(q) + \varepsilon_j(q) - \varepsilon_k(q), \\ 4dk = \varepsilon_o(q) - \varepsilon_i(q) - \varepsilon_j(q) + \varepsilon_k(q). \end{cases} \tag{1.6}$$

Считая умножения на степени двойки более элементарной операцией по сравнению с вещественным умножением, мы не будем учитывать их при анализе вычислительной сложности рассматриваемых алгоритмов.

Определим число вещественных умножений, необходимых для перемножения двух кватернионов. Умножение комплексных чисел может быть реализовано по схеме "три умножения, три сложения", тогда, в соответствии с представлением (1.3), умножение двух кватернионов общего вида может быть реализовано с помощью девяти вещественных умножений. Пусть далее $s = \alpha + \beta i$ - i-кватернион; $t = \gamma + \delta j$ - j-кватернион. Тогда для вычисления произведений sq и qt необходимо по шесть вещественных умножений, а для одновременного вычисления произведения sqt - девять вещественных умножений:

$$\begin{aligned} sq = &\big((\alpha - \beta)b + \alpha(a - b)\big) + \big((\alpha - \beta)b + \beta(a + b)\big) + \\ &+ \big((\alpha - \beta)d + \alpha(c - d)\big)j + \big((\alpha - \beta)d + \alpha(c + d)\big)k; \end{aligned} \tag{1.7}$$

$$sqt = \Big(\Big[(\alpha-\beta)(b-d) + \alpha(a-b-c+d) \Big] \delta + \Big[(\alpha-\beta)b + \alpha(a-b) \Big] (\gamma-\delta) \Big) +$$
$$+ \Big(\Big[(\alpha-\beta)(b-d) + \beta(a+b-c-d) \Big] \delta + \Big[(\alpha-\beta)b + \beta(a+b) \Big] (\gamma-\delta) \Big) i +$$
$$+ \Big(\Big[(\alpha-\beta)(b-d) + \alpha(a-b-c+d) \Big] \delta + \Big[(\alpha-\beta)d + \alpha(c-d) \Big] (\gamma-\delta) \Big) j +$$
$$+ \Big(\Big[(\alpha-\beta)(b-d) + \beta(a+b-c-d) \Big] \delta + \Big[(\alpha-\beta)d + \beta(c+d) \Big] (\gamma-\delta) \Big) k. \tag{1.8}$$

При этом считаем, что произведения и суммы констант $\alpha, \beta, \gamma, \delta$ выполнены заранее.

1.2 Представление данных в кодах Гамильтона-Эйзенштейна

Пусть H - алгебра кватернионов, определенная в предыдущем параграфе, кватернионы γ_1 и γ_2 - примитивные корни третьей степени из единицы, лежащие в различных экземплярах поля комплексных чисел $C_1 = C(i)$ и $C_2 = C(j)$, каноническим образом вложенных в H:

$$\gamma_1 = \exp\left\{\frac{2\pi i}{3}\right\}, \quad \gamma_2 = \exp\left\{\frac{2\pi j}{3}\right\}$$

кватернионы $\overline{\gamma}_1$ и $\overline{\gamma}_2$ - соответствующие образы в H элементов, сопряженных в C_1 и C_2 элементам γ_1 и γ_2 :

$$\overline{\gamma}_1 = \exp\left\{-\frac{2\pi i}{3}\right\}, \quad \overline{\gamma}_2 = \exp\left\{-\frac{2\pi j}{3}\right\}.$$

Пусть $q = (p + qi + rj + sk) \in H$ и существуют $a, b, c, d \in R$ такие, что

$$q = (a\gamma_1 + b\overline{\gamma}_1)\gamma_2 + (c\gamma_1 + d\overline{\gamma}_1)\overline{\gamma}_2 .$$

Тогда четверку вещественных чисел (a, b, c, d) называют кодом Гамильтона-Эйзенштейна кватерниона q и обозначают <q>.

Переход от кода Гамильтона-Эйзенштейна к алгебраической форме кватерниона имеет вид

$$q=\frac{\sqrt{3}}{4}\begin{pmatrix}a & b & c & d\end{pmatrix}\begin{pmatrix}1/\sqrt{3} & -1 & -1 & \sqrt{3}\\ 1/\sqrt{3} & 1 & -1 & -\sqrt{3}\\ 1/\sqrt{3} & -1 & 1 & -\sqrt{3}\\ 1/\sqrt{3} & 1 & 1 & \sqrt{3}\end{pmatrix}\begin{pmatrix}1\\ i\\ j\\ k\end{pmatrix}. \qquad (1.9)$$

То есть

$$p=\tfrac{1}{4}(a+b+c+d)$$
$$q=\tfrac{\sqrt{3}}{4}(-a+b-c+d)$$
$$r=\tfrac{\sqrt{3}}{4}(-a-b+c+d)$$
$$s=\tfrac{3}{4}(a-b-c+d)$$

Кватернионы специального вида имеют следующие коды.

Элемент алгебры C_1, простое комплексное число, корни w_1:

$$<q_0+q_1 i>=(a,b,a,b)\text{, где } q_0=\frac{1}{2}(a+b),\quad q_1=\frac{\sqrt{3}}{2}(b-a).$$

Элемент алгебры C_2, комплексное число с j, корни w_2:

$$<q_0+q_2 j>=(a,a,c,c)\text{, где } q_0=\frac{1}{2}(a+c),\quad q_1=\frac{\sqrt{3}}{2}(c-a).$$

Гамма-элементы:

$$<\gamma_1>=(-1,0,-1,0),\ <\overline{\gamma}_1>=(0,-1,0,-1),$$
$$<\gamma_2>=(-1,-1,0,0),\ <\overline{\gamma}_2>=(0,0,-1,-1).$$

Вещественные числа:

$$q=a\in \mathrm{R}\text{, то } <q>=(a,a,a,a).$$

Пусть $q=(a,b,c,d)$, $s\in \mathrm{C}_1$, $t\in \mathrm{C}_2$, $<s>=(\alpha,\beta,\alpha,\beta)$, $<t>=(\sigma,\sigma,\tau,\tau)$. Тогда умножения кватерниона q общего вида на i- или j-кватернионы требуют не более шести нетривиальных вещественных умножений и шести вещественных сложений (здесь считается, что сложения компонент кодов i- и j-кватернионов выполнены заранее):

$$<sq>=\begin{matrix}\big((\beta-\alpha)(a-b)+\alpha a,\ (\beta-\alpha)(a-b)+\beta b,\\ (\beta-\alpha)(c-d)+\alpha c,\ (\beta-\alpha)(c-d)+\beta d\big)\end{matrix};$$

$$<qt>=\begin{matrix}\big((\tau-\sigma)(a-c)+\sigma a,\ (\tau-\sigma)(b-d)+\sigma b,\\ (\tau-\sigma)(a-c)+\tau c,\ (\tau-\sigma)(b-d)+\tau d\big)\end{matrix}.$$

В частности, умножения кватернионов q общего вида на γ_1, $\overline{\gamma}_1$, γ_2 или $\overline{\gamma}_2$ требуют только двух вещественных сложений:

$$\begin{matrix}<\gamma_1 q>=(-b,\ a-b,\ -d,\ c-d), & <\overline{\gamma}_1 q>=(b-a,\ -a,\ d-c,\ -c),\\ <q\gamma_2>=(-c,\ -d,\ a-c,\ b-d), & <q\overline{\gamma}_2>=(c-a,\ d-b,\ -a,\ -b).\end{matrix} \qquad (1.10)$$

Непосредственное последовательное умножение кватерниона общего вида на i- и j-кватернионы требует 12 вещественных умножений. Одновременное выполнение такой пары умножений требует 9 вещественных умножений и 15 вещественных сложений:

$$\begin{matrix}<sqt>=\big((\tau-\sigma)\big[(\beta-\alpha)(d-c-b+a)-\alpha(c-a)\big]-\sigma(\beta-\alpha)(b-a)+\sigma\alpha a,\\ (\tau-\sigma)\big[(\beta-\alpha)(d-c-b+a)-\beta(d-b)\big]-\sigma(\beta-\alpha)(b-a)+\sigma\beta a,\\ (\tau-\sigma)\big[(\beta-\alpha)(d-c-b+a)-\alpha(c-a)\big]-\tau(\beta-\alpha)(d-c)+\tau\alpha c,\\ (\tau-\sigma)\big[(\beta-\alpha)(d-c-b+a)-\beta(d-b)\big]-\tau(\beta-\alpha)(d-c)+\tau\beta c\big).\end{matrix}$$

Автоморфизмы (1.5) тела кватернионов H над P индуцируют следующие преобразования кодов. Пусть $<q>=(a,b,c,d)$, тогда

$$<\varepsilon_i(q)>=(c,d,a,b),\quad <\varepsilon_j(q)>=(b,a,d,c),\quad <\varepsilon_k(q)>=(d,c,b,a),$$

и переход от кватерниона к его автоморфному образу реализуется в кодах тривиально (без вещественных умножений).

1.3 Алгебра гиперкомплексных чисел

Коммутативной гиперкомплексной алгеброй B будем называть четырехмерную ассоциативную алгебру над P

$$\mathbf{B} = \{h = a + bi + cj + dij;\ a,b,c,d \in \mathbf{R}\} \tag{1.11}$$

с определяющими соотношениями для умножений базисных элементов $\{1, i, j, ij\}$:

$$i^2 = j^2 = -1, \qquad ij = ji, \quad (ij)^2 = 1. \tag{1.12}$$

Операция сложения в данной алгебре осуществляется покомпонентно. Соотношение (1.12) для умножения базисных элементов индуцирует правило умножения произвольных элементов алгебры B

$$\begin{aligned} hq &= (a + bi + cj + dij)(x + yi + zj + wij) = \\ &= (ax - by - cz + wd) + \\ &+ (ay + bx - cw - dz)i + \\ &+ (az - bw + cx - dy)j + \\ &+ (aw + bz + cy + dx)ij. \end{aligned} \tag{1.13}$$

Отметим еще два полезных свойства алгебры B:

- поле комплексных чисел X канонически вкладывается в B:

$$a + bi \to a + bi + 0 \cdot j + 0 \cdot ij$$

- справедливы соотношения

$$h = a + bi + cj + dij = (a + bi) + (c + di)j = s_0 + s_1 j, \tag{1.14}$$

$$h = a + bi + cj + dij = (a + bi) + (d - ci)ij = z_0 + z_1 \varepsilon, \quad (\varepsilon = ij;\ \varepsilon^2 = 1).$$

Непосредственное умножение элементов алгебры B по формуле (1.13) требует 16 вещественных умножений и 12 вещественных сложений. Как обычно, будем считать, что:

- при оценке вычислительной сложности один из сомножителей предполагается постоянным и, следовательно, все арифметические операции над его компонентами могут быть реализованы заранее;

- умножения на степени числа 2 являются более простыми операциями, чем сложения и умножения, и не учитываются при анализе вычислительной сложности алгоритмов ДПФ.

Пусть $h = a + bi + cj + dij$ - элемент алгебры B общего вида; $s = x + yi$ - комплексное число, являющееся константой в контексте рассматриваемых алгоритмов. В соответствии с представлением (1.14) вычисление произведения hs эквивалентно выполнению двух комплексных умножений. При использовании схемы "три умножения, три сложения" искомое произведение принимает вид

$$\begin{aligned} ht = &\Big(\big(a+b\big)x - b\big(x+y\big)\Big) + \\ &+\Big(\big(a+b\big)x - a\big(x-y\big)\Big)i + \\ &+\Big(\big(c+d\big)x - d\big(x+y\big)\Big)j + \\ &+\Big(\big(c+d\big)x - c\big(x-y\big)\Big)ij \end{aligned} \qquad (1.15)$$

и вычисляется посредством шести вещественных умножений и шести вещественных сложений.

Одновременное умножение на два комплексных корня в этой алгебре лучше выполнять по очереди.

Непосредственной проверкой легко убедиться также в справедливости следующего утверждения.

Отображения

$$\begin{cases} \varepsilon_o(h) = a + bi + cj + dij \\ \varepsilon_i(h) = a + bi - cj - dij \\ \varepsilon_j(h) = a - bi + cj - dij \\ \varepsilon_k(h) = a - bi - cj + dij \end{cases} \qquad (1.16)$$

сохраняют сумму и произведение элементов алгебры B, действуют тождественно на P (то есть являются *автоморфизмами* B *над* P).

1.4 Прямая сумма комплексных алгебр

Известно, что коммутативно-ассоциативная гиперкомплексная алгебра B (1.11)-(1.12) изоморфна прямой сумме комплексных алгебр:

$$\mathrm{B} = \mathrm{C} \oplus \mathrm{C} .$$

Для перехода от стандартного представления гиперкомплексного числа

$$h = a + bi + cj + dij \tag{1.17}$$

к его представлению в прямой сумме комплексных алгебр, выполним замену переменных:

$$u_0 = 1 + j \ , \ u_1 = 1 - j \ , \ u_2 = i + ij \ , \ u_3 = i - ij \ . \tag{1.18}$$

В этом случае произвольный элемент алгебры примет вид

$$h = \frac{1}{2}\left((a+c)u_0 + (a-c)u_1 + (b+d)u_2 + (b-d)u_3 \right).$$

Таблица умножения новых базисных элементов имеет вид, показанный ниже.

Таблица 1.1. Правила умножения новых базисных элементов

Базисные элементы	u_0	u_1	u_2	u_3
u_0	$2u_0$	0	$2u_2$	0
u_1	0	$2u_1$	0	$2u_3$
u_2	$2u_2$	0	$-2u_0$	0
u_3	0	$2u_3$	0	$-2u_1$

В таком представлении произведение двух произвольных элементов алгебры

$$\left(xu_0 + yu_1 + zu_2 + vu_3\right)\left(au_0 + bu_1 + cu_2 + du_3\right)$$

разбивается на два независимых произведения, подобных произведениям комплексных чисел

$$\left(xu_0 + zu_2\right)\left(\alpha u_0 + \gamma u_2\right) = 2\left((x\alpha - z\gamma)u_0 + (x\gamma + z\alpha)u_2 \right),$$

$$\left(yu_1 + vu_3\right)\left(\beta u_1 + \delta u_3\right) = 2\left((y\beta - v\delta)u_1 + (y\delta + v\beta)u_3 \right),$$

и требует 6 вещественных умножений и 6 вещественных сложений.

Все свойства гиперкомплексной алгебры в таком представлении сохраняются.

2 ДВУМЕРНОЕ ДПФ И ЕГО БЫСТРЫЕ АЛГОРИТМЫ В СПЕЦИАЛЬНЫХ АЛГЕБРАИЧЕСКИХ СТРУКТУРАХ

2.1 Определение двумерного ДПФ в алгебре комплексных чисел и в четырехмерных алгебрах

Пусть $x(n_1, n_2)$ - входной массив размера $N \times N$, тогда его комплексный спектр Фурье

$$\tilde{X}(m_1, m_2) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x(n_1, n_2) \omega^{n_1 m_1 + n_2 m_2}, \quad 0 \le m_1, m_2 \le N-1, \qquad (2.1)$$

где $\omega = \exp\left\{2\pi i / N\right\}$ - комплексный корень из единицы степени N.

Под дискретным преобразованием Фурье со специальным представлением данных в рамках курсового проекта будем понимать преобразование вида

$$X(m_1, m_2) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} \omega_1^{m_1 n_1} x(n_1, n_2) \omega_2^{m_2 n_2}. \qquad (2.2)$$

Основная идея такого преобразования заключается в погружении входного сигнала и корней ω_k в четырехмерную алгебру **A** (в рамках курсового проекта это одна из алгебраических структур, описанных в разделе 1). При этом корни $\omega_1 = \exp\left\{2\pi i / N\right\}$, $\omega_2 = \exp\left\{2\pi j / N\right\}$ лежат в разных экземплярах поля комплексных чисел, вложенных в А, у них разные мнимые единицы - i и j.

Отметим, что умножения на степени ω_1 и ω_2 в соотношении (2.2) записаны слева и справа от входного сигнала. Это сделано для унификации записи. При использовании алгебры кватернионов Н это принципиально, так как умножение в ней некоммутативно. При использовании гиперкомплексной алгебры В порядок сомножителей значения не имеет.

Поскольку элемент алгебры A (кватернион $q = a + bi + cj + dk$ или гиперкомплексное число $h = a + bi + cj + dij$) определяется набором четырех вещественных чисел (a, b, c, d), то комплексный спектр (2.1) может быть получен из спектра (2.2) в алгебре A следующим образом:

$$\tilde{X}(m_1, m_2) = X(m_1, m_2)\mathbf{LI},$$

где $X(m_1, m_2) = (\chi_0(m_1, m_2), \chi_1(m_1, m_2), \chi_2(m_1, m_2), \chi_3(m_1, m_2))$ - вектор компонент спектра,

$$\mathbf{L} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad \mathbf{I} = \begin{pmatrix} 1 \\ i \end{pmatrix}.$$

Таким образом, мультипликативная сложность вычисления $\tilde{X}(m_1, m_2)$ совпадает со сложностью вычисления спектра в алгебре A, т.к. умножения на матрицы **L**, **I** не требуют выполнения нетривиальных операций вещественного умножения.

Далее излагаются схемы декомпозиции двумерного дискретного преобразования Фурье (2.2) и способы его вычисления.

2.2 Алгоритм двумерного ДПФ по основанию 2

Пусть $N = 2^k$. Представим (2.2) в виде четырех сумм, разделяя входную последовательность по четным и нечетным значениям каждого индекса n_1, n_2:

$$X(m_1,m_2)=\sum_{n_1,n_2=0}^{N-1}\omega_1^{m_1n_1}x(n_1,n_2)\omega_2^{m_2n_2}=$$

$$=\sum_{a,b=0}^{1}\omega_1^{am_1}\sum_{n_1,n_2=0}^{N/2-1}\left(\omega_1^2\right)^{m_1n_1}x_{ab}(n_1,n_2)\left(\omega_2^2\right)^{m_2n_2}\omega_2^{bm_2}= \qquad (2.3)$$

$$=\sum_{a,b=0}^{1}\omega_1^{am_1}X_{ab}(m_1,m_2)\omega_2^{bm_2},$$

где

$$x_{ab}(n_1,n_2)=x(2n_1+a,2n_2+b),$$

$$X_{ab}(m_1,m_2)=\sum_{n_1,n_2=0}^{N/2-1}\left(\omega_1^2\right)^{m_1n_1}x_{ab}(n_1,n_2)\left(\omega_2^2\right)^{m_2n_2},\ 0\le m_1,m_2\le N/2-1.$$

Вычисление спектра для остальных значений пар (m_1,m_2) производится без дополнительных умножений и может быть записано в матричной форме

$$\begin{bmatrix}X(m_1,m_2)\\X(m_1+N/2,m_2)\\X(m_1,m_2+N/2)\\X(m_1+N/2,m_2+N/2)\end{bmatrix}=\begin{bmatrix}1&1&1&1\\1&-1&1&-1\\1&1&-1&-1\\1&-1&-1&1\end{bmatrix}\bullet\begin{bmatrix}X_{00}(m_1,m_2)\\\omega_1^{m_1}X_{10}(m_1,m_2)\\X_{01}(m_1,m_2)\omega_2^{m_2}\\\omega_1^{m_1}X_{11}(m_1,m_2)\omega_2^{m_2}\end{bmatrix}$$

или в следующем виде

$$X\left(m_1+r\frac{N}{4},m_2+s\frac{N}{4}\right)=\sum_{a,b=0}^{1}(-1)^{ar}\omega_1^{am_1}X_{ab}(m_1,m_2)\omega_2^{bm_2}(-1)^{bs}, \quad (2.4)$$

$$r,s=0,1\ .$$

2.3 Алгоритм двумерного ДПФ по основанию 4

Разобьем теперь входную последовательность на 16 подпоследовательностей и запишем (2.2) в виде

$$X(m_1, m_2) = \sum_{a,b=0}^{3} \omega_1^{am_1} X_{ab}(m_1, m_2) \omega_2^{bm_2}, \qquad (2.5)$$

где

$$x_{ab}(n_1, n_2) = x(4n_1 + a, 4n_2 + b), \qquad (2.6)$$

$$X_{ab}(m_1, m_2) = \sum_{n_1, n_2=0}^{N/4-1} \left(\omega_1^4\right)^{m_1 n_1} x_{ab}(n_1, n_2) \left(\omega_2^4\right)^{m_2 n_2}, \; 0 \le m_1, m_2 \le N/4 - 1.$$

При этом значения спектра для остальных значений аргументов вычисляются без дополнительных умножений, а именно:

$$X\left(m_1 + r\frac{N}{4}, m_2 + s\frac{N}{4}\right) = \sum_{a,b=0}^{3} i^{ar} \omega_1^{am_1} X_{ab}(m_1, m_2) \omega_2^{bm_2} j^{bs}, \qquad (2.7)$$

$$r, s = 0, 1, 2, 3.$$

Умножения на степени базовых элементов i и j тривиальны, они сводятся к перестановкам элементов кода и/или смене знака компонент.

2.4 Алгоритм с расщеплением основания

Рассмотрим еще одну схему декомпозиции двумерного спектра (2.2), в которой ДПФ объема $N \times N$ сводится к ДПФ $N/2 \times N/2$ элементов входной последовательности с четными индексами и двенадцати ДПФ объемом $N/4 \times N/4$ с элементами, имеющими хотя бы один нечетный индекс. Пусть

$$A = \{(0,1), (0,3), (1,0), (1,1), (1,2), (1,3), (2,1), (2,3), (3,0), (3,1), (3,2), (3,3)\},$$

тогда

$$X(m_1,m_2)=\sum_{n_1,n_2=0}^{N/2-1}\left(\omega_1^2\right)^{m_1n_1}x(2n_1,2n_2)\left(\omega_2^2\right)^{m_2n_2}+$$
$$+\sum_{(a,b)\in A}\omega_1^{am_1}\sum_{n_1,n_2=0}^{N/4-1}\left(\omega_1^4\right)^{m_1n_1}x_{ab}(n_1,n_2)\left(\omega_2^4\right)^{m_2n_2}\omega_2^{bm_2}, \qquad (2.8)$$

где $x_{ab}(n_1,n_2)$ определены в (2.6). При этом по-прежнему справедливо соотношение (2.7) для остальных значений m_1,m_2.

2.5. Двумерное ДПФ по основанию 3

Пусть $N=3^r$. Константы ω_1,ω_2 из (2.2) будем считать заданными кодами Гамильтона-Эйзенштейна. Спектр (2.2) записывается в форме

$$X(m_1,m_2)=\sum_{a,b=0}^{2}\omega_1^{am_1}X_{ab}(m_1,m_2)\,\omega_2^{bm_2},$$

где

$$X_{ab}(m_1,m_2)=\sum_{n_1,n_2=0}^{N/3-1}\omega_1^{3n_1m_1}x(3n_1+a,3n_2+b)\,\omega_2^{3n_2m_2}.$$

Значения $X_{ab}(m_1,m_2)$ достаточно полностью вычислять для пар $(m_1,m_2)\in\Delta$, где Δ - "фундаментальная область":

$$\Delta=\left\{(\mu,\upsilon):0\le\mu,\upsilon\le N/3-1\right\}.$$

Значения $X(m_1,m_2)$ для пар (m_1,m_2), лежащих в областях, полученных из Δ аддитивными сдвигами на векторы

$$a=\left(r\frac{N}{3},s\frac{N}{3}\right),\ r,s=0,1,2,$$

не требуют для вычисления дополнительных вещественных умножений и определяются из соотношения

$$X\left(m_1 + r\frac{N}{3}, m_2 + s\frac{N}{3}\right) = \sum_{a,b=0}^{2} \gamma_1^{ar} \left(\omega_1^{am_1} X_{ab}\left(m_1, m_2\right) \omega_2^{bm_2}\right) \gamma_2^{bs} \ , \quad (2.9)$$

где константы γ_1, γ_2 определены в разделе 1.2. При использовании представления данных в кодах Гамильтона-Эйзенштейна умножение на степени этих констант не требует нетривиальных умножений и не повышает мультипликативную сложность алгоритма.

2.6 Двумерное ДПФ по основанию 6

Пусть $N = 6^r$, константы ω_1, ω_2 по-прежнему заданы кодами Гамильтона-Эйзенштейна. Спектр (2.2) может быть записан в форме

$$X\left(m_1, m_2\right) = \sum_{a,b=0}^{5} \omega_1^{am_1} X_{ab}\left(m_1, m_2\right) \omega_2^{bm_2} \ ,$$

где

$$X_{ab}\left(m_1, m_2\right) = \sum_{n_1, n_2=0}^{N/6-1} \omega_1^{6n_1 m_1} x\left(6n_1 + a, 6n_2 + b\right) \omega_2^{6n_2 m_2} \ .$$

Пусть Δ - “фундаментальная область”:

$$\Delta = \left\{\left(\mu, \upsilon\right) : 0 \le \mu, \upsilon \le N/6 - 1\right\} .$$

Значения $X\left(m_1, m_2\right)$ для пар $\left(m_1, m_2\right)$, лежащих в областях, полученных из Δ аддитивными сдвигами на векторы

$$a = \left(r\frac{N}{6}, s\frac{N}{6}\right), r, s = 0, 1, 2, 3, 4, 5 \ ,$$

не требуют для вычисления дополнительных вещественных умножений и определяются из соотношения

$$X\left(m_1 + r\frac{N}{6}, m_2 + s\frac{N}{6}\right) = \sum_{a,b=0}^{5} \left(-\gamma_1\right)^{ar} \left(\omega_1{}^{am_1} X_{ab}\left(m_1, m_2\right) \omega_2{}^{bm_2}\right) \left(-\gamma_2\right)^{bs} . \tag{2.10}$$

2.7 Реализация: от коротких длин к большим

Как видно из параграфов 2.2-2.6, все схемы декомпозиции показывают идею быстрых алгоритмов двумерного дискретного преобразования Фурье, которая заключается в сведении преобразования заданной длины к нескольким преобразованиям меньшего объема. Однако на практике реализация преобразования выполняется в обратном порядке от минимальной длины к максимальной, как и в одномерном случае.

Известно, что для реализации схемы декомпозиции по основанию p на первом шаге необходимо выполнить все преобразования размером $p \times p$, выбирая для этого элементы входного массива, лежащие на расстоянии N/p. Далее из полученных частичных спектров формируются спектры размером $p^2 \times p^2$ с использованием соотношений (2.4), (2.7) (2.9) или (2.10). И так далее, вплоть до формирования искомого спектра размером $N \times N$, где $N = p^k$.

Однако в этом случае, как известно из курса "Теория цифровой обработки сигналов", выходные отсчеты оказываются переставленными в двоично-инверсном порядке или другим аналогичным способом в зависимости от основания алгоритма p. Чтобы получить на выходе правильный порядок отсчетов, а также для удобства выбора значений частичных спектров из общей матрицы данных (алгоритм реализуется "на месте"), рекомендуется выполнить перестановку данных в начале работы. Расположение отсчетов до и после перестановки проиллюстрировано на рис.2.1.

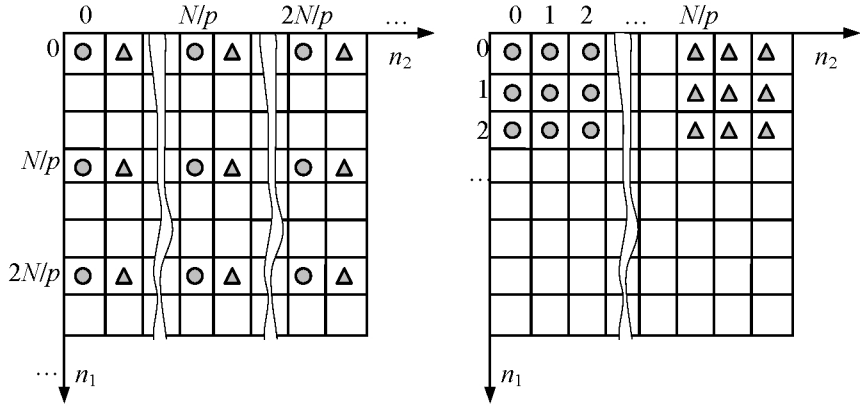

Рис.2.1. Расположение отсчетов в массиве до и после перестановки

Таким образом, алгоритм вычисления преобразования (2.2) окончательно принимает следующий вид:

1. Перестановка элементов входного массива в порядке p-ичной инверсии (см. Приложение 4).

2. Расчет всех преобразований размером $p \times p$. Данные для них теперь лежат в массиве входных данных подряд.

3. В цикле по длине преобразования, вычисляемого на данном шаге, выполнять переход от каждых p^2 частичных спектров текущего размера $l \times l$ к спектру размером $lp \times lp$ до тех пор, пока не будет достигнут размер входного массива $N \times N$ (см. рис. 2.2, 2.3).

2.8 Учет вещественности входного сигнала

Первоначально такое преобразование (2.2) использовалось как вспомогательное для эффективного вычисления преобразования (2.1) при вещественном входном сигнале. Существуют два способа учета вещественности входного сигнала, которые отражены в темах курсового проекта. Это - синтез совмещенных алгоритмов и учет симметрий спектра в рассматриваемых алгебрах. Остановимся на этих способах подробнее.

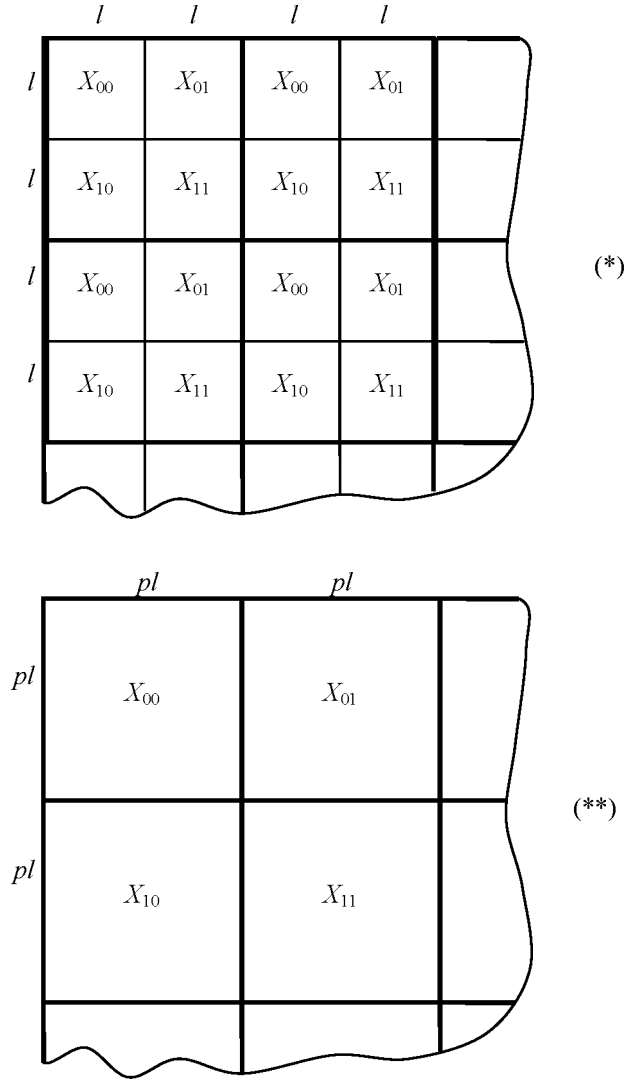

Рис.2.2. Расположение частичных спектров на текущем (*) и последующем (**) шаге для p=2

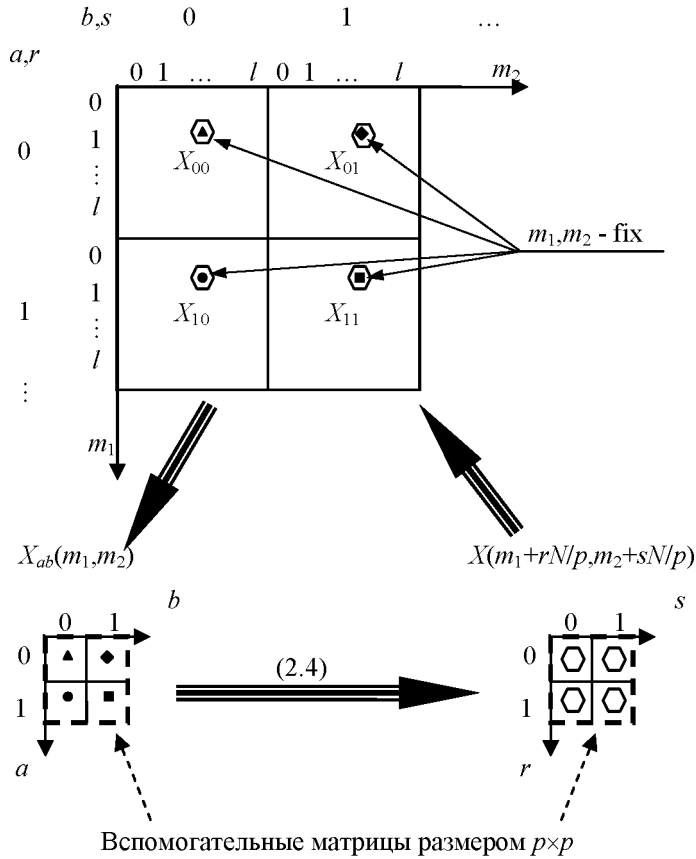

Рис. 2.3. Схема пересчета частичных спектров "на месте"

Как и везде в этой главе, мы будем оперировать с некоторой алгеброй A, понимая, что это может быть как алгебра кватернионов H (в том числе в форме кодов Гамильтона-Эйзенштейна), так и алгебра гиперкомплексных чисел B (в том числе в форме прямой суммы комплексных алгебр) каждая со своим набором автоморфизмов.

2.8.1 Алгоритм двумерного ДПФ с совмещением

Пусть $x(n_1, n_2)$ вещественная N-периодическая по каждому аргументу функция; N - четное. Ее спектр Фурье

$$X(m_1, m_2) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x(n_1, n_2) \omega^{n_1 m_1 + n_2 m_2} = \sum_{a,b=0}^{1} \omega^{am_1 + bm_2} S_{ab}(m_1, m_2), \tag{2.11}$$

где

$$S_{ab}(m_1, m_2) = \sum_{n_1, n_2=0}^{N/2-1} x(2n_1 + a, 2n_2 + b) \left(\omega^2\right)^{m_1 n_1 + m_2 n_2}. \tag{2.12}$$

Используем для представления входных данных функцию $q(n_1, n_2)$ со значениями в теле кватернионов H:

$$q(n_1, n_2) = x(2n_1, 2n_2) + x(2n_1, 2n_2 + 1)i + x(2n_1 + 1, 2n_2)j + x(2n_1 + 1, 2n_2 + 1)k, \tag{2.13}$$

или в гиперкомплексной алгебре B:

$$q(n_1, n_2) = x(2n_1, 2n_2) + x(2n_1, 2n_2 + 1)i + x(2n_1 + 1, 2n_2)j + x(2n_1 + 1, 2n_2 + 1)ij. \tag{2.14}$$

Вычисление спектра исходной последовательности можно свести к вычислению спектра новой последовательности иной внутренней структуры, но меньшего объема, что и делает алгоритм более быстрым.

Спектр $Q(m_1, m_2)$ такой последовательности определяется равенством

$$Q(m_1, m_2) = \sum_{n_1, n_2=0}^{N/2-1} \left(\omega_1{}^2\right)^{m_1 n_1} q(n_1, n_2) \left(\omega_2{}^2\right)^{m_2 n_2}.$$

Выделить "частичные спектры" $S_{ab}(m_1, m_2)$ из массива, являющегося результатом ДПФ в алгебре A, можно путем решением следующей системы уравнений:

$$\begin{cases} 4S_{00}(m_1,m_2) = \\ \quad = Q(m_1,m_2) + \varepsilon_i\left(Q(m_1,m_2)\right) + \varepsilon_j\left(Q(-m_1,-m_2)\right) + \varepsilon_k\left(Q(-m_1,-m_2)\right), \\ 4iS_{01}(m_1,m_2) = \\ \quad = Q(m_1,m_2) + \varepsilon_i\left(Q(m_1,m_2)\right) - \varepsilon_j\left(Q(-m_1,-m_2)\right) - \varepsilon_k\left(Q(-m_1,-m_2)\right), \\ 4jS_{10}(m_1,m_2) = \\ \quad = Q(m_1,m_2) - \varepsilon_i\left(Q(m_1,m_2)\right) + \varepsilon_j\left(Q(-m_1,-m_2)\right) - \varepsilon_k\left(Q(-m_1,-m_2)\right), \\ 4kS_{11}(m_1,m_2) = \\ \quad = Q(m_1,m_2) - \varepsilon_i\left(Q(m_1,m_2)\right) - \varepsilon_j\left(Q(-m_1,-m_2)\right) + \varepsilon_k\left(Q(-m_1,-m_2)\right), \end{cases} \tag{2.15}$$

где $\varepsilon_i, \varepsilon_j, \varepsilon_k$ - автоморфизмы соответствующей алгебры (1.6) или (1.15).

Для решения данной системы требуется не более 4 вещественных умножений на степени двойки.

Комплексные значения отсчетов спектра двумерного ДПФ в области $0 \le m_1, m_2 \le N/2$ находятся непосредственным применением формулы (2.11).

Вычисление спектра для остальных значений пар (m_1, m_2) производится без дополнительных умножений и может быть представлено в матричной форме следующим образом:

$$\begin{bmatrix} X(m_1,m_2) \\ X(m_1+N/2,m_2) \\ X(m_1,m_2+N/2) \\ X(m_1+N/2,m_2+N/2) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_{00}(m_1,m_2) \\ \omega^{m_1} S_{10}(m_1,m_2) \\ \omega^{m_2} S_{01}(m_1,m_2) \\ \omega^{m_1+m_2} S_{11}(m_1,m_2) \end{bmatrix}. \tag{2.16}$$

2.8.2 Учет симметрий спектра вещественного сигнала

Основное свойство спектра вещественного сигнала в рассматриваемых алгебрах - это его симметрия. Так, спектр вещественного сигнала удовлетворяет следующим свойствам симметрии:

$$\begin{aligned} X\left(N-m_1,m_2\right)&=\varepsilon_j\left(X\left(m_1,m_2\right)\right),\\ X\left(m_1,N-m_2\right)&=\varepsilon_i\left(X\left(m_1,m_2\right)\right),\\ X\left(N-m_1,N-m_2\right)&=\varepsilon_k\left(X\left(m_1,m_2\right)\right), \end{aligned} \qquad (2.17)$$

где $\varepsilon_i,\varepsilon_j,\varepsilon_k$ - это по-прежнему автоморфизмы соответствующей алгебры (1.6) или (1.15).

Такое свойство позволяет на каждом шаге алгоритма (см. п.2.7) производить вычисления по формулам (2.4), (2.7) (2.9) или (2.10) и т.п. в области размером $\left(p^k/2\times p^k/2\right)$ вместо области размером $\left(p^k\times p^k\right)$ (см. рис.2.4), где p - основание алгоритма, k - номер шага декомпозиции. Недостающие отсчеты заполняются на основании свойств (2.17) в соответствии со схемой, показанной на рис. 2.5.

2.9 Оценка сложности алгоритмов

В рамках курсового проекта необходимо определить и сравнить теоретические вычислительные сложности рассматриваемых алгоритмов. Ниже приводится методика расчета и сложности рассмотренных алгоритмов двумерного ДПФ.

2.9.1 Методика оценки вычислительной сложности быстрых алгоритмов двумерного ДПФ

Из курса "Теория цифровой обработки сигналов" известна методика расчета вычислительной сложности быстрых алгоритмов одномерного ДПФ. Она полностью распространяется на двумерный случай. В общем случае для расчета сложности любого из описанных алгоритмов достаточно знать основание декомпозиции p (или структуру алгоритма в случае расщепления основания) и сложность базовых операций в используемой алгебраической структуре.

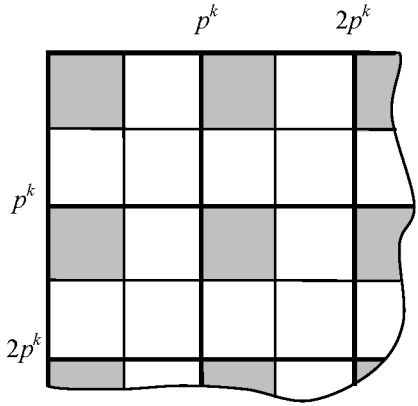

Рис.2.4. Области, которые достаточно рассчитать по формулам (2.4), (2.7), (2.9), (2.10)

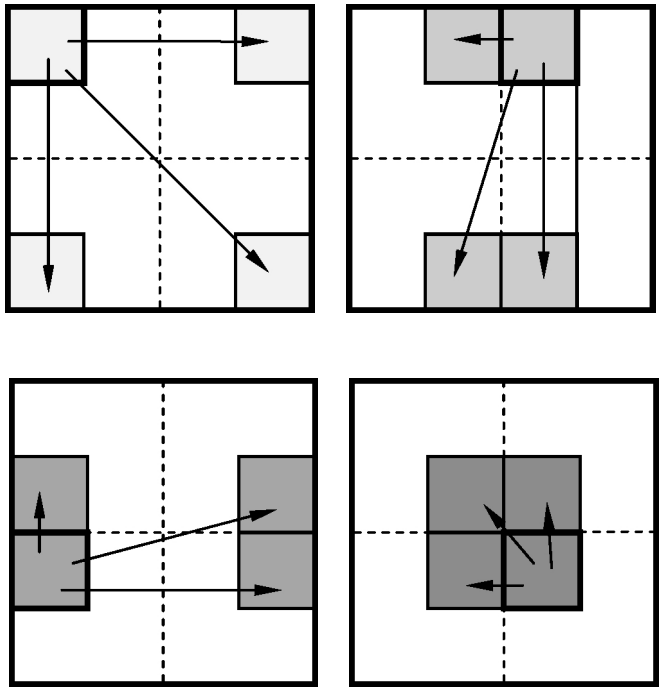

Рис. 2.5. Заполнение недостающих областей на основании свойств симметрии

Обозначим $M^*_{\mathbf{A}}$, $A^*_{\mathbf{A}}$ - количество операций вещественного умножения и сложения для операции умножения в алгебре А, $A^+_{\mathbf{A}}$ - количество вещественных сложений для операции сложения в алгебре А. Учтем, что в некоторых алгебраических структурах сложность умножения двух произвольных элементов алгебры может отличаться от сложности умножения такого элемента на комплексное число (то есть на степени корней ω_k). Поэтому для второго случая введем дополнительное обозначение $M^*_{\mathbf{AC}}$, $A^*_{\mathbf{AC}}$.

Тогда вычислительная сложность быстрого алгоритма двумерного ДПФ "по основанию p" описывается следующими рекуррентными соотношениями:

$$M\left(N\times N\right)=p^2M\left(\frac{N}{p}\times\frac{N}{p}\right)+\left(p-1\right)^2\left(\frac{N}{p}\right)^2M^*_{\mathbf{A}}+2\left(p-1\right)\left(\frac{N}{p}\right)^2M^*_{\mathbf{AC}},$$

$$A\left(N\times N\right)=p^2A\left(\frac{N}{p}\times\frac{N}{p}\right)+\left(p-1\right)^2\left(\frac{N}{p}\right)^2A^*_{\mathbf{A}}+2\left(p-1\right)\left(\frac{N}{p}\right)^2A^*_{\mathbf{AC}}+ \quad (2.18)$$
$$+p^3\left(\frac{N}{p}\right)^2A^+_{\mathbf{A}},$$

Методика решения таких соотношений следующая. Известно, что сложность алгоритмов типа Кули-Тьюки логарифмическая. Пусть тогда

$$M\left(N\times N\right)=C_MN^2\log_pN+D_MN^2,$$
$$A\left(N\times N\right)=C_AN^2\log_pN+D_AN^2. \quad (2.19)$$

Для определения первой пары констант C_M, C_A отбросим второе слагаемое в (2.19), а логарифмический член подставим в (2.18). Получим

$$C_MN^2\log_pN=p^2C_M\left(\frac{N}{p}\right)^2\left(\log_pN-1\right)+\left(p-1\right)^2\left(\frac{N}{p}\right)^2M^*_{\mathbf{A}}+$$
$$+2\left(p-1\right)\left(\frac{N}{p}\right)^2M^*_{\mathbf{AC}},$$

$$C_A N^2 \log_p N = p^2 C_A \left(\frac{N}{p}\right)^2 \left(\log_p N - 1\right) + \left(p-1\right)^2 \left(\frac{N}{p}\right)^2 A^*_{\mathbf{A}} +$$
$$+ 2\left(p-1\right)\left(\frac{N}{p}\right)^2 A^*_{\mathbf{AC}} + p^3 \left(\frac{N}{p}\right)^2 A^+_{\mathbf{A}},$$

откуда определяются константы:

$$C_M = \left(\frac{p-1}{p}\right)^2 M^*_{\mathbf{A}} + 2\frac{(p-1)}{p^2} M^*_{\mathbf{AC}},$$
$$C_A = \left(\frac{p-1}{p}\right)^2 A^*_{\mathbf{A}} + 2\frac{(p-1)}{p^2} A^*_{\mathbf{AC}} + pA^+_{\mathbf{A}}. \qquad (2.20)$$

Для определения второй пары констант вручную вычисляют количество операций для вычисления преобразования наименьшего размера $p \times p$. Как правило, оно выполняется без умножений, а количество вещественных сложений обозначим a_p. Подставляя $N = p$ в (2.19), получим

$$0 = C_M p^2 + D_M p^2,$$
$$a_p = C_A p^2 + D_A p^2.$$

Откуда окончательно находим

$$D_M = -C_M,$$
$$D_A = a_p / p^2 - C_A. \qquad (2.21)$$

Определим теперь вычислительные сложности некоторых из рассмотренных алгоритмов в соответствующих алгебраических структурах.

2.9.2 Сложности алгоритмов по основанию 2 и 4 в алгебре кватернионов

Пусть в качестве алгебры, в которой реализовано двумерное дискретное преобразование Фурье, выступает алгебра кватернионов H. Как показано в п.1.1, сложность операций в этой алгебре составляет:

$$M^*_{\mathbf{A}} = 9;\ A^*_{\mathbf{A}} = 15;\ M^*_{\mathbf{AC}} = 6;\ A^*_{\mathbf{AC}} = 6;\ A^+_{\mathbf{A}} = 4. \qquad (2.22)$$

Тогда, без учета вещественности входного сигнала, оценки сложности описанных алгоритмов при p=2, 4 (см. п.2.2., 2.3) приведены в таблице 2.1.

Таблица 2.1. Сложность алгоритмов кватернионного ДПФ по основанию p

p	Мультипликативная сложность	Аддитивная сложность
2	$M\left(N\times N\right)\le\frac{21}{4}N^2\log_2 N-\frac{21}{4}N^2$	$A\left(N\times N\right)\le\frac{59}{4}N^2\log_2 N-\frac{27}{4}N^2$
4	$M\left(N\times N\right)\le\frac{117}{32}N^2\log_2 N-\frac{117}{16}N^2$	$A\left(N\times N\right)\le\frac{427}{32}N^2\log_2 N-\frac{235}{16}N^2$

При использовании одного из способов учета вещественности входного сигнала теоретическая сложность алгоритма уменьшается приблизительно в 4 раза.

Оценка сложности алгоритма с расщеплением основания строится подобным образом, но для нее рекуррентные соотношения принимают вид

$$M\left(N\times N\right)=M\left(\frac{N}{p}\times\frac{N}{p}\right)+2p\left(2p-1\right)M\left(\frac{N}{2p}\times\frac{N}{2p}\right)+\left(\left(2p-1\right)^2-1\right)\left(\frac{N}{2p}\right)^2M_{\mathbf{A}}^{*}+4\left(p-1\right)\left(\frac{N}{2p}\right)^2M_{\mathbf{AC}}^{*},$$

$$A\left(N\times N\right)=A\left(\frac{N}{p}\times\frac{N}{p}\right)+2p\left(2p-1\right)A\left(\frac{N}{2p}\times\frac{N}{2p}\right)+\left(\left(2p-1\right)^2-1\right)\left(\frac{N}{2p}\right)^2A_{\mathbf{A}}^{*}+4\left(p-1\right)\left(\frac{N}{2p}\right)^2A_{\mathbf{AC}}^{*}+p^3\left(\frac{N}{p}\right)^2A_{\mathbf{A}}^{+},\quad(2.18)$$

где p - это большее основание (в п. 2.4 p=2).

2.9.3 Сложности алгоритмов в гиперкомплексной алгебре

Пусть на этот раз в качестве алгебры, в которой реализовано двумерное дискретное преобразование Фурье, выступает гиперкомплексная алгебра B. Как показано в п.1.3, сложность операций в этой алгебре составляет:

$$M_{\mathbf{A}}^{*}=6;\ A_{\mathbf{A}}^{*}=14;\ M_{\mathbf{AC}}^{*}=6;\ A_{\mathbf{AC}}^{*}=6;\ A_{\mathbf{A}}^{+}=4.$$

Тогда, без учета вещественности входного сигнала, оценки сложности тех же алгоритмов при p=2, 4 приведены в таблице 2.2.

Таблица 2.2. Сложность алгоритмов гиперкомплексного ДПФ по основанию p

p	Мультипликативная сложность	Аддитивная сложность
2	$M\left(N\times N\right)\le\frac{9}{2}N^2\log_2 N-\frac{9}{2}N^2$	$A\left(N\times N\right)\le\frac{29}{2}N^2\log_2 N-\frac{13}{2}N^2$
4	$M\left(N\times N\right)\le\frac{45}{16}N^2\log_2 N-\frac{45}{8}N^2$	$A\left(N\times N\right)\le\frac{209}{16}N^2\log_2 N-\frac{113}{8}N^2$

2.9.4 Сложности алгоритмов с представлением данных в кодах Гамильтона-Эйзенштейна

Как было показано в п.1.2, коды Гамильтона-Эйзенштейна - это только удобный в ряде случаев способ представления кватернионов. Поэтому сложность арифметических операций в кодах равна сложности соответствующих операций в алгебре кватернионов (2.22). Такое представление данных в рамках курсового проекта используется при синтезе алгоритмов двумерного ДПФ по основаниям 3 и 6. Оценки их вычислительной сложности представим в таблице 2.3.

Таблица 2.3. Сложность алгоритмов ДПФ по основанию p
с представлением данных в кодах Гамильтона-Эйзенштейна

p	Мультипликативная сложность	Аддитивная сложность
3	$M\left(N\times N\right)\le\frac{20}{3}N^2\log_3 N-\frac{20}{3}N^2$	$A\left(N\times N\right)\le\frac{64}{3}N^2\log_3 N-\frac{176}{9}N^2$
6	$M\left(N\times N\right)\le\frac{95}{12}N^2\log_6 N-\frac{95}{12}N^2$	$A\left(N\times N\right)\le\frac{433}{12}N^2\log_6 N-\frac{1261}{36}N^2$

2.9.5 Сложности алгоритмов в прямой сумме комплексных алгебр

Сложность операций в прямой сумме комплексных алгебр (см.п.1.4) отличается от сложности тех же операций в гиперкомплексной алгебре только тем, что умножение двух произвольных элементов требует шести вещественных умножений и шести вещественных сложений. Соответственно изменятся только аддитивные сложности алгоритмов, как показано в таблице 2.4.

Таблица 2.4. Сложность алгоритмов ДПФ по основанию p
с представлением данных в прямой сумме комплексных алгебр

p	Мультипликативная сложность	Аддитивная сложность

2	$M\left(N\times N\right)\le\frac{9}{2}N^2\log_2 N-\frac{9}{2}N^2$	$A\left(N\times N\right)\le\frac{25}{2}N^2\log_2 N-\frac{9}{2}N^2$
4	$M\left(N\times N\right)\le\frac{45}{16}N^2\log_2 N-\frac{45}{8}N^2$	$A\left(N\times N\right)\le\frac{174}{16}N^2\log_2 N-\frac{77}{8}N^2$

2.9.6 Сравнительный анализ сложности алгоритмов

На рис. 2.6 показана зависимость количества операций вещественного сложения и умножения от размера преобразования $N\times N$ для декомпозиции по основанию 2. Очевидно, что наименьшие вычислительные затраты обеспечивает представление данных в прямой сумме комплексных алгебр.

На рисунке 2.7 представлено сравнение вычислительной сложности преобразования в прямой сумме комплексных алгебр при разных основаниях алгоритма. Очевидно, что с ростом p количество операций уменьшается. Усложнение структуры декомпозиции (расщепление основания) так же снижает вычислительную сложность, однако здесь это не отражено.

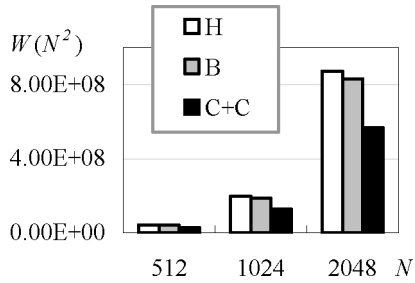

Рис.2.6. Суммарное количество операций двумерного ДПФ в разных алгебрах

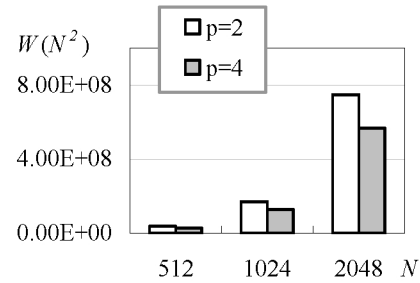

Рис.2.7. Суммарное количество операций для вычисления двумерного ДПФ в прямой сумме комплексных алгебр по основанию p

3 ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ ДВУМЕРНОГО ДПФ

Основная проблема при реализации многомерных (в том числе двумерных) преобразований – это увеличение объема вычислений с ростом размерности. Одним из наиболее естественных путей решения этой проблемы в условиях быстрого развития методов и алгоритмов высокопроизводительной обработки информации, является параллельная реализация преобразования (2.2). Ниже приведены два подхода: распараллеливание вычислений в рамках многомерной схемы Кули-Тьюки и использование структуры алгебры B (1.11).

Кроме того, на основании оценок вычислительной сложности двумерного ДПФ в гиперкомплексной алгебре и прямой сумме комплексных алгебр, полученных в главе 2, определяется предполагаемое ускорение и эффективность параллельных алгоритмов. Показано, что структура алгебры позволяет достичь 90% эффективности распараллеливания.

Материал этой главы является дополнительным, может быть использован при выполнении УИРС и дипломов по соответствующей тематике. В рамках курсового проекта задача реализации параллельного алгоритма не ставится.

3.1 Распараллеливание по структуре декомпозиции

Рассмотрим возможность параллельной реализации вычисления двумерного ДПФ (2.2) для схемы декомпозиции ДПФ "по основанию 2" (см. п.2.2). Напомним, что основное соотношение по аналогии с БПФ Кули-Тьюки имеет вид

$$X(m_1, m_2) = \sum_{a,b=0}^{1} \omega_1^{am_1} X_{ab}(m_1, m_2) \omega_2^{bm_2}, \qquad (3.1)$$

где $0 \le m_1, m_2 \le N/2 - 1$,

$$X_{ab}(m_1, m_2) = \sum_{n_1, n_2 = 0}^{N/2-1} \left(\omega_1^2\right)^{m_1 n_1} x(2n_1 + a, 2n_2 + b) \left(\omega_2^2\right)^{m_2 n_2}.$$

В параграфе 2.8 показано, что умножения на $\omega_1^{am_1}$, $\omega_2^{bm_2}$ достаточно выполнять только для $0 \le m_1, m_2 \le N/4 - 1$, так как остальные значения гиперкомплексного спектра могут быть найдены с использованием автоморфизмов одной из используемых алгебр.

Ключевой операцией в таком алгоритме является реконструкция (3.1) полного спектра $X(m_1, m_2)$ по известным (найденным) значениям частичных спектров $X_{ab}(m_1, m_2)$ размером $N/2 \times N/2$. Предположим, что каждый частичный спектр будет рассчитан на отдельном процессоре, причем время работы процессоров будет примерно одинаковым (так как одинаковы размеры массивов вычисляемых спектров и алгоритм их вычисления). Далее при выполнении ключевой операции загрузка процессоров будет выглядеть, как это показано на рис. 3.1. Необходимо отметить, что простой процессоров II, III будет небольшим, так как сложность умножения на один корень не намного меньше сложности одновременного умножения на оба корня сразу, а в случае гиперкомплексной алгебры эти сложности совпадают.

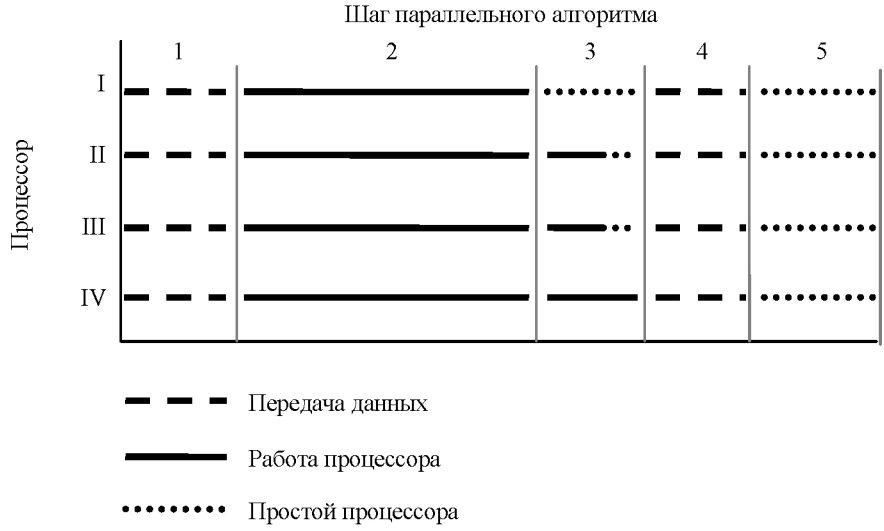

Рис. 3.1. Схема работы и простоя процессоров

В соответствии с рис.3.1 параллельный алгоритм состоит из следующих шагов.

Шаг 1. Распределение данных по процессорам.

Шаг 2. Расчет частичных спектров на каждом процессоре.

Шаг 3. Умножение элементов частичных спектров на степени ω_1, ω_2.

Шаг 4. Передача результатов на один процессор.

Шаг 5. Объединение спектров по формуле (3.1).

Ясно, что таким способом процесс может быть распараллелен на любое число процессоров, кратное четырем, за счет нескольких шагов декомпозиции типа (3.1). Предполагаемое время вычисления спектра обратно пропорционально числу процессоров, так как основные вычислительные затраты приходятся на расчет набора ДПФ уменьшенного размера. В случае размерности данных $d>2$ может быть применена аналогичная схема с распараллеливанием на каждом шаге на 2^d процессов.

Также можно воспользоваться другими схемами декомпозиции, формируя параллельные алгоритмы по тем же принципам.

3.2 Распараллеливание за счет структуры алгебры

Такой способ может быть применен только при использовании представления данных в ассоциативно-коммутативной гиперкомплексной алгебре. Основан он на переходе к представлению данных в прямой сумме комплексных алгебр:

$$\mathbf{B} \cong \mathbf{C} \oplus \mathbf{C} . \qquad (3.2)$$

Рассмотрим принципы формирования такого параллельного алгоритма. Пусть выполнена замена переменных, описанная в п.1.4. Повторим ее здесь для удобства изложения. Итак, представим произвольный элемент гиперкомплексной алгебры B в форме (1.17)

$$h = a + bi + cj + dij$$

и выполним замену переменных (1.18):

$$u_0 = 1 + j \ , \ u_1 = 1 - j \ , \ u_2 = i + ij \ , \ u_3 = i - ij \ .$$

В новом представлении гиперкомплексное число h запишется в виде

$$h = \frac{1}{2}\left((a+c)u_0 + (a-c)u_1 + (b+d)u_2 + (b-d)u_3\right).$$

Очевидно, что переход к новому представлению потребует четырех вещественных сложений. Однако для вещественных (входной сигнал) и комплексных (корни ω_k) чисел этот переход не потребует выполнения нетривиальных арифметических операций.

Обратный переход к исходному представлению также требует четырех вещественных сложений на отсчет гиперкомплексного спектра.

Таблица умножения новых базисных элементов приведена в п.1.4. Часть произведений новых базисных элементов u_k в ней равна нулю. На этом факте и строится параллельный алгоритм. Конкретно будут равны нулю произведения элементов u_0 и u_2 с элементами u_1 и u_3. Это означает, что вычисление произведения двух произвольных гиперкомплексных чисел состоит в таком представлении из двух совершенно независимых частей. Вместо произведения

$$\left(xu_0 + yu_1 + zu_2 + vu_3\right)\left(\alpha u_0 + \beta u_1 + \gamma u_2 + \delta u_3\right)$$

теперь достаточно независимо вычислить два произведения:

$$\left(xu_0 + zu_2\right)\left(\alpha u_0 + \gamma u_2\right) = 2\left(\left(x\alpha - z\gamma\right)u_0 + \left(x\gamma + z\alpha\right)u_2\right),$$
$$\left(yu_1 + vu_3\right)\left(\beta u_1 + \delta u_3\right) = 2\left(\left(y\beta - v\delta\right)u_1 + \left(y\delta + v\beta\right)u_3\right).$$

В процессе вычисления ДПФ необходимо выполнять две ключевых операции. Это умножение гиперкомплексного числа на степени корней ω_1, ω_2 и сложение гиперкомплексных чисел. Таким образом, наиболее трудоемкая операция быстрого алгоритма двумерного ДПФ – вычисление произведения гиперкомплексных чисел - может быть распараллелена на две независимые ветви, не требующие обмена данными. Это означает, что время выполнения операции будет снижено практически в два раза.

Структура последовательного быстрого алгоритма ДПФ (см., например, п.2.2) такова, что при использовании такого представления возможно полное разделение вычислений по тому же принципу. В результате мы приходим к следующему параллельному алгоритму двумерного ДПФ с представлением данных в алгебре гиперкомплексных чисел.

Шаг 1. Переход от исходного представления к новому (тривиально, так как входные данные вещественные, а корни - комплексные).

Шаг 2. Распределение данных по двум процессорам.

Шаг 3. Расчет преобразования с использованием алгоритмов типа Кули-Тьюки.

Шаг 4. Реконструкция гиперкомплексного спектра.

Загрузка процессоров при реализации такого параллельного алгоритма показана на рис.3.2. Здесь отсутствует первый тривиальный шаг алгоритма.

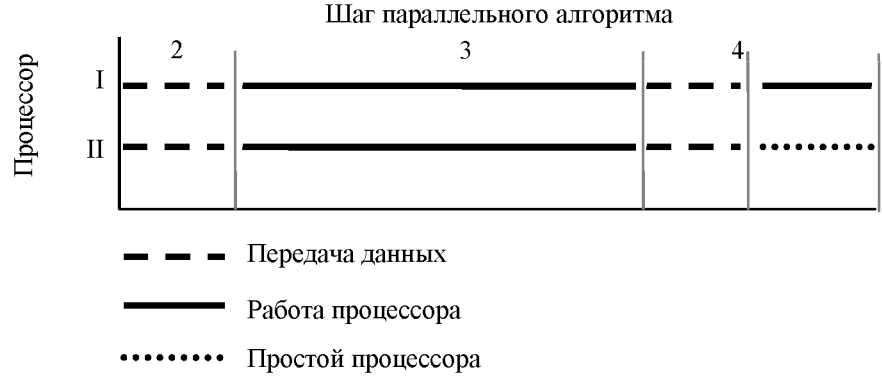

Рис. 3.2. Схема работы и простоя процессоров: 2 - распределение данных по процессорам, 3 - расчет спектров, 4 – объединение и реконструкция спектра

3.3 Распараллеливание гиперкомплексного дискретного преобразования Фурье в многомерном случае

Под многомерным гиперкомплексным дискретным преобразованием Фурье (ГДПФ) будем понимать преобразование вида

$$X\left(m_1,\ldots,m_d\right)=\sum_{n_1,\ldots,n_d=0}^{N-1} x\left(n_1,\ldots,n_d\right)W^{<\mathbf{m},\mathbf{n}>}, \qquad (3.3)$$

$$W^{<\mathbf{m},\mathbf{n}>}=\prod_{k=1}^{d}\omega_k^{m_k n_k},\ \omega_k^N=1,$$

где по-прежнему комплексные корни ω_k лежат в разных экземплярах комплексной алгебры, вложенных в 2^d-мерную коммутативно-ассоциативную гиперкомплексную алгебру $\mathbf{B}_d$.

3.3.1 *Многомерная алгебра* $\mathbf{B}_d$

Определение. 2^d-мерная **P**-algebra $\mathbf{B}_d$ с базисом

$$\Lambda = \left\{ \prod_{i \in I} \varepsilon_i^{\alpha_i},\ \alpha_i \in \{0,1\};\ I = \{1,\ldots,d\} \right\}$$

называется коммутативно-ассоциативной гиперкомплексной алгеброй. Здесь $\varepsilon_1,\ \varepsilon_2,\ \ldots \varepsilon_d$ - базис d-мерного векторного пространства **V**, $\varepsilon_i^0 = 1,\ \varepsilon_i^1 = \varepsilon_i$, а их правило умножения задается соотношениями

$$\varepsilon_i \varepsilon_j = \varepsilon_j \varepsilon_i, \quad \varepsilon_i^2 = \beta_i, \quad i, j \in I,\ \beta_i = \pm 1.$$

Произвольный элемент $g \in \mathbf{B}_d$ имеет вид

$$g = \xi_0 E_0 + \ldots + \xi_{2^d-1} E_{2^d-1} = \sum_{t \in T} \xi_t E_t\ , \tag{3.4}$$

где

$$E_t = \prod_{i \in I} \varepsilon_i^{\alpha_i}\ ,\quad t = \sum_{i \in I} \alpha_i 2^{i-1} \in T = \left\{0,1,\ldots,2^d - 1\right\}. \tag{3.5}$$

В этой алгебре сложение элементов выполняется покомпонентно, а умножение полностью определяется правилами умножения базисных элементов:

$$E_t E_\tau = \Psi(t,\tau) E_{t \oplus \tau}, \quad \forall t, \tau \in T\ ,$$

где $\oplus$ обозначает побитное сложение по модулю 2,

$$\Psi(t,\tau) = \prod_{i \in I} \beta_i^{h_i(t,\tau)}\ ,\ h_i(t,\tau) = \alpha_i \alpha_i'\ , \tau = \sum_{i \in I} \alpha_i' 2^{i-1}\ .$$

Здесь мы рассматриваем только такую гиперкомплексную алгебру, которая изоморфна прямой сумме комплексных алгебр:

$$\mathbf{B}_d \cong \underbrace{\mathbf{C} \dotplus \mathbf{C} \dotplus \ldots \dotplus \mathbf{C}}_{2^{d-1}}\ ,$$

что гарантирует минимальную сложность операции умножения в $\mathbf{B}_d$. В этом случае по крайней мере одному элементу α_i соответствует $\beta_i = -1$. Без потери общности будем считать, что $\beta_1 = -1$ и $\beta_i = 1$ для остальных $i \in I$. Такая структура алгебры и позволяет нам строить параллельный алгоритм..

3.3.2 Базовый последовательный алгоритм многомерного ГДПФ

Основное соотношение по аналогии с двумерным случаем имеет вид

$$X(m_1,...,m_d) = \sum_{a_1,...,a_d=0}^{1} X_{a_1...a_d}(m_1,...,m_d) W^{<\mathbf{a},\mathbf{m}>}, \qquad (3.6)$$

где

$$X_{a_1...a_d}(m_1,...,m_d) = \\ = \sum_{n_1,...,n_d=0}^{N/2-1} x(2n_1 + a_1,...,2n_d + a_d)\left(W^2\right)^{<\mathbf{m},\mathbf{n}>},$$

$$0 \le m_j \le N/2 - 1, \quad j = \overline{1,d}.$$

3.3.3 Параллельный алгоритм, основанный на схеме декомпозиции

В быстром алгоритме вычисления многомерного гиперкомплексного спектра ключевой является операция (3.6). Как и в двумерном случае (см. п.3.1), возможно сделать параллельным расчет частичных спектров $X_{a_1...a_d}(m_1,...,m_d)$ на любом шаге декомпозиции. Очевидно, что таким способом процесс может быть распараллелен на 2^d компьютеров на каждрм шаге декомпозиции. Однако чем больше шагов мы сделаем, тем меньше будет доля времени на выполнение параллельной части и, соответственно, ниже эффективность распараллеливания.

3.3.4 Параллельный алгоритм, основанный на структуре алгебры

По аналогии с двумерным случаем (п.3.2) построим параллельный алгоритм для вычислений в многомерной гиперкомплексной алгебре. Пусть произвольный элемент

h 2^d-мерной алгебры $\mathbf{B}_d$ определен соотношением (3.4). Разобьем множество $\{E_t\}_{t\in T}$ на две части: $t\in T'$, если E_t не включает в себя ε_1, и $t\in T''$ в противном случае. Учитывая выбранный способ нумерации базисных элементов, в первую часть войдут базисные элементы E_t с четными индексами t, а во вторую часть - с нечетными. Введем замену переменных:

$$\mathbf{U}_0 = \mathbf{A}\,\mathbf{E}_0\,,\ \mathbf{U}_1 = \mathbf{A}\,\mathbf{E}_1\,, \tag{3.7}$$

где

$$\mathbf{E}_0 = \{E_t\}_{t\in T'},\ \mathbf{E}_1 = \{E_t\}_{t\in T''},$$

$$\mathbf{U}_0 = \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{2^{d-1}} \end{pmatrix},\ \mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & -1 \\ 1 & 1 & -1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & -1 & 1 & \cdots & -1 \end{pmatrix}.$$

Каждый столбец и каждая строка матрицы $\mathbf{A}$ содержит ровно 2^{d-2} отрицательных значений из 2^{d-1} значений.

Правила умножения новых базисных элементов имеют вид

$$u_j^2 = \begin{cases} pu_j, & \text{при} \quad j < p, \\ -pu_{j-2^{d-1}} & \text{при} \quad j \ge p, \end{cases}$$

$$u_j u_k = \begin{cases} pu_k, & \text{if } k = j + p, \\ 0, & \text{в противном случае,} \end{cases}$$

где $p = 2^{d-1}$. Отметим, что многие произведения равны нулю. Этот факт позволяет нам представить произведение двух произвольных элементов алгебры $\mathbf{B}_d$ в следующем виде

$$g \cdot s = \sum_{t \in T} \xi_t E_t \cdot \sum_{\tau \in T} \varsigma_\tau E_\tau = \sum_{t \in T} a_t u_t \cdot \sum_{\tau \in T} b_\tau u_\tau =$$
$$= \sum_{t \in T'} \left(a_t u_t + a_{t+p} u_{t+p} \right) \left(b_t u_t + b_{t+p} u_{t+p} \right).$$

Таким образом, произведение двух произвольных элементов алгебры $\mathbf{B}_d$ сводится к p независимым произведениям комплексных чисел, каждое из которых требует трех вещественных умножений и трех вещественных сложений. Итак, произведение может быть распараллелено на p ветвей, а сложение, в силу линейности замены переменных, сохраняет покомпонентную форму.

Пример: d=3. Произвольный элемент восьмимерной гиперкомплексной алгебры $\mathbf{B}_3$ имеет вид

$$z = \xi_0 + \xi_1 \varepsilon_1 + \xi_2 \varepsilon_2 + \xi_3 \varepsilon_1 \varepsilon_2 +$$
$$+ \xi_4 \varepsilon_3 + \xi_5 \varepsilon_1 \varepsilon_3 + \xi_6 \varepsilon_2 \varepsilon_3 + \xi_7 \varepsilon_1 \varepsilon_2 \varepsilon_3 ,$$

где $\varepsilon_1^2 = -1, \varepsilon_2^2 = \varepsilon_3^2 = 1$. Замена переменных:

$$u_0 = 1 + \varepsilon_2 + \varepsilon_3 + \varepsilon_2 \varepsilon_3 ,$$
$$u_1 = 1 + \varepsilon_2 - \varepsilon_3 - \varepsilon_2 \varepsilon_3 ,$$
$$u_2 = 1 - \varepsilon_2 + \varepsilon_3 - \varepsilon_2 \varepsilon_3 ,$$
$$u_3 = 1 - \varepsilon_2 - \varepsilon_3 + \varepsilon_2 \varepsilon_3 ,$$
$$u_4 = \varepsilon_1 + \varepsilon_1 \varepsilon_2 + \varepsilon_1 \varepsilon_3 + \varepsilon_1 \varepsilon_2 \varepsilon_3 ,$$
$$u_5 = \varepsilon_1 + \varepsilon_1 \varepsilon_2 - \varepsilon_1 \varepsilon_3 - \varepsilon_1 \varepsilon_2 \varepsilon_3 ,$$
$$u_6 = \varepsilon_1 - \varepsilon_1 \varepsilon_2 + \varepsilon_1 \varepsilon_3 - \varepsilon_1 \varepsilon_2 \varepsilon_3 ,$$
$$u_7 = \varepsilon_1 - \varepsilon_1 \varepsilon_2 - \varepsilon_1 \varepsilon_3 + \varepsilon_1 \varepsilon_2 \varepsilon_3 .$$

Тогда

$$z = \frac{1}{4} \left(\left(\xi_0 + \xi_2 + \xi_4 + \xi_6 \right) u_0 + \left(\xi_0 + \xi_2 - \xi_4 - \xi_6 \right) u_1 + \right.$$
$$+ \left(\xi_0 - \xi_2 + \xi_4 - \xi_6 \right) u_2 + \left(\xi_0 - \xi_2 - \xi_4 + \xi_6 \right) u_3 +$$
$$+ \left(\xi_1 + \xi_3 + \xi_5 + \xi_7 \right) u_4 + \left(\xi_1 + \xi_3 - \xi_5 - \xi_7 \right) u_5 +$$
$$\left. + \left(\xi_1 - \xi_3 + \xi_5 - \xi_7 \right) u_6 + \left(\xi_1 - \xi_3 - \xi_5 + \xi_7 \right) u_7 \right).$$

Правило умножения базисных элементов показано в таблице 3.1.

Таблица 3.1. Правило умножения базисных элементов (d=3)

Базисные элементы	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7
u_0	$4u_0$	0	0	0	$4u_4$	0	0	0
u_1	0	$4u_1$	0	0	0	$4u_5$	0	0
u_2	0	0	$4u_2$	0	0	0	$4u_6$	0
u_3	0	0	0	$4u_3$	0	0	0	$4u_7$
u_4	$4u_4$	0	0	0	$-4u_0$	0	0	0
u_5	0	$4u_5$	0	0	0	$-4u_1$	0	0
u_6	0	0	$4u_6$	0	0	0	$-4u_2$	0
u_7	0	0	0	$4u_7$	0	0	0	$-4u_3$

Вместо вычисления произведения

$$\sum_{j=0}^{7} x_j u_j \sum_{k=0}^{7} y_k u_k$$

теперь достаточно вычислить 4 независимых произведения:

$$\left(x_0u_0 + x_4u_4\right)\left(y_0u_0 + y_4u_4\right),$$
$$\left(x_1u_1 + x_5u_5\right)\left(y_1u_1 + y_5u_5\right),$$
$$\left(x_2u_2 + x_6u_6\right)\left(y_2u_2 + y_6u_6\right),$$
$$\left(x_3u_3 + x_7u_7\right)\left(y_3u_3 + y_7u_7\right).$$

Таким образом, в этом случае алгоритм может быть распараллелен на 4 ветви.

3.3.5 Оценка предполагаемой эффективности параллельных алгоритмов многомерного ГДПФ

Для теоретической оценки эффективности параллельной программы необходимо задаться некоторыми значениями следующих стандартных характеристик:

T_{op} - время выполнения одной вещественной операции на одном процессоре ;

T_l - время латентности при пересылке данных;

T_m - время пересылки одного байта информации.

Возьмем для расчетов характеристики, представленные на сайте www.parallel.ru для кластера вычислительного центра МГУ:

$$T_{op} = 4мкс\ ,\ T_l = 5.6мкс\ ,\ T_m = 0.012мкс\ .$$

Вычислительная сложность последовательного алгоритма ГДПФ

Из литературы известно, что вычислительная сложность последовательного алгоритма ГДПФ равна:

$$M_B\left(N^d\right) \le \left(1 - 1/2^d\right) N^d \log_2 N + O\left(N^d\right) \qquad (3.8)$$

$$A_B\left(N^d\right) \le \frac{2d}{2^d} \cdot N^d \log_2 N + O\left(N^d\right), \qquad (3.9)$$

где $M_B\left(N^d\right)$ - количество гиперкомплексных умножений, $A_B\left(N^d\right)$ - количество гиперкомплексных сложений, d - размерность входного сигнала. Учитывая сложность гиперкомплексного умножения

$$M_* = 3 \cdot 2^{d-1},\ \ A_* = \left(4d - 1\right) \cdot 2^{d-1}$$

и сложения

$$A_+ = 2^d\ ,$$

получим суммарную сложность последовательного алгоритма:

$$G\left(N^d\right) = M_B\left(N^d\right) \cdot \left(M_* + A_*\right) + A_B\left(N^d\right) \cdot A_+ \le$$
$$\le \left(d \cdot 2^{d+1} + 2^d - 1\right) N^d \log_2 N + O\left(N^d\right).$$

Следовательно, предполагаемое время его выполнения равно:

$$T\left(N^d\right) \approx G\left(N^d\right) \cdot T_{op} \le \left(d \cdot 2^{d+1} + 2^d - 1\right) N^d \log_2 N \cdot T_{op}\ .$$

Как показано выше, метод позволяет распараллелить алгоритм по $p = 2^{d-1}$ процессорам. В этом случае затраты параллельного алгоритма состоят из следующих частей. Для преобразования гиперкомплексных данных в новое представление и обратно необходимо 4^{d-1} вещественных сложений на каждый входной элемент. Но для вещественного входного сигнала и комплексных корней такой переход тривиален. Значит, всего на переходы от исходного представлений к новому и обратно потребуется $4^{d-1}N^d$ сложений.

Далее данные должны быть разосланы по процессорам, а затем собраны обратно. Объем пересылаемых данных составляет $2N^d \cdot 4$ байт для каждого шага.

Наконец, на каждом процессоре должен быть выполнен последовательный алгоритм со своими двумя компонентами. При этом сложность операций умножения и сложения составит $M_*^c = 3$, $A_*^c = 3$, $A_+^c = 2$.

Таким образом, предполагаемое время работы параллельного алгоритма примет вид

$$T_p^1\left(N^d\right) \approx 4^{d-1}N^d \cdot T_{op} + 2\left(T_l + 8N^d T_m\right) +$$
$$+ M_B\left(N^d\right)\left(M_*^c + A_*^c\right)\cdot T_{op} + A_B\left(N^d\right)A_+^c \cdot T_{op} \approx$$
$$\approx \left(\frac{\left(3\cdot 2^{d+1} + 4d - 6\right)}{2^d}\log_2 N + 4^{d-1}\right)N^d T_{op} + 2\left(T_l + 8N^d T_m\right).$$

В таблице 3.2 показано ожидаемое ускорение

$$U = T\left(N^d\right)\Big/T_p^1\left(N^d\right)$$

и эффективность

$$E = T\left(N^d\right)\Big/pT_p^1\left(N^d\right)$$

алгоритма, где p - это количество процессоров. Ожидаемая эффективность чрезвычайно высока.

Таблица 3.2. Ожидаемое ускорение и эффективность первого метода распараллеливания

N	d=2, p=2		d=3, p=4	
	U	E	U	E
256	1.72	0.86	2.50	0.62
512	1.75	0.87	2.61	0.65
1024	1.77	0.88	2.70	0.68
2048	1.79	0.89	2.78	0.70
4096	1.80	0.90	2.86	0.71

Эффективность распараллеливания на основе декомпозиции

Этот способ распараллеливания требует $p = \left(2^d\right)^s$ процессоров, где s-количество шагов или глубина декомпозиции. Каждый процессор вычисляет частичные спектры размером $\left(N/2^s\right)^d$. Кроме того, необходимо восстанавливать из них полный спектр (3.3), что требует трех гиперкомплексных умножений на элемент спектра. С учетом этого предполагаемое время работы алгоритма составит:

$$T_p^2\left(N^d\right) \approx \left(\left(\frac{d\cdot 2^{d+1}-2^d-1}{2^{sd}}\right)\left(\log_2 N - s\right) + 3\cdot 2^{d-2}\left(2d+1\right)s\right)\cdot N^d \cdot T_{op} +$$

$$+2\left(T_l + 4\cdot 2^d\left(\frac{N}{2^s}\right)^d T_m\right).$$

Ускорение и эффективность алгоритма показаны в таблице 3.3.

Таблица 3.3. Ожидаемое ускорение и эффективность второго метода распараллеливания

N	d=2, p=4		d=3, p=8	
	U	E	U	E
16	1,87	0,47	2,74	0,34
32	2,10	0,52	3,16	0,39
64	2,28	0,57	3,51	0,44
128	2,43	0,61	3,82	0,48
256	2,55	0,64	4,09	0,51
512	2,66	0,66	4,32	0,54

Здесь эффективность уменьшается с ростом размерности сигнала. Эффективность этого метода существенно ниже эффективности первого подхода. Однако комбинированный алгоритм позволит достичь наибольшего ускорения на доступном количестве процессоров.

Иллюстрация предполагаемого времени выполнения алгоритмов

Построим зависимость предполагаемого времени работы алгоритма от размера преобразования для случаев перечисленных в таблице 3.4 для двумерного случая и в таблице 3.5 для d=3. Здесь p - количество процессоров. Соответствующие графики представлены на рис. 3.3 и 3.4 (T_p - время выполнения программы).

Таблица 3.4. Список алгоритмов для d=2

p	Алгоритм
1	Последовательный алгоритм
2	Алгоритм на основе свойств алгебры
4	Алгоритм на основе декомпозиции (1 шаг)
8	Комбинированный алгоритм
16	Алгоритм на основе декомпозиции (2 шага)

Таблица 3.5. Список алгоритмов для d=3

p	Алгоритм
1	Последовательный алгоритм
4	Алгоритм на основе свойств алгебры

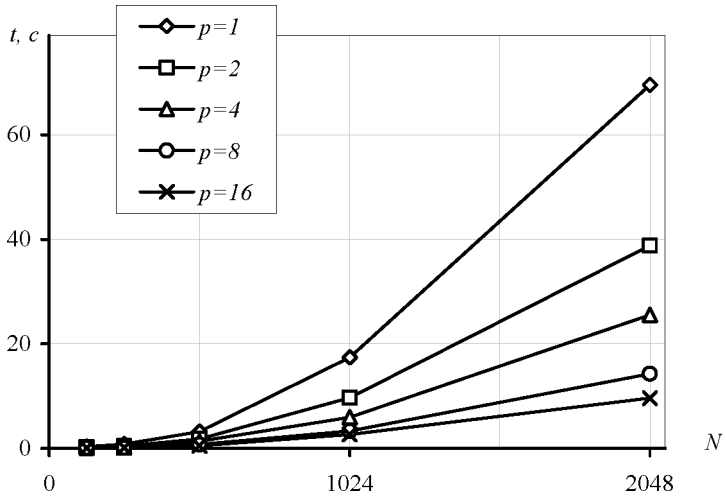

Рис. 3.3. Время выполнения двумерного ГДПФ последовательным и параллельными алгоритмами

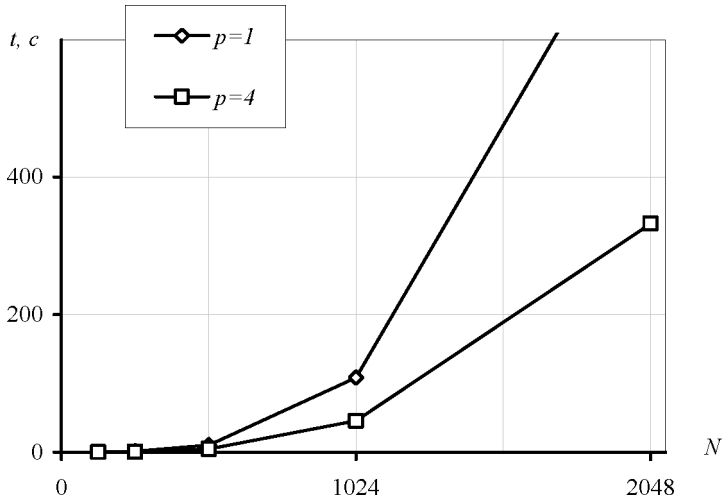

Рис. 3.4. Время выполнения трехмерного ГДПФ последовательным и параллельным алгоритмами

4 ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ КУРСОВОГО ПРОЕКТА

В этом разделе приводятся требования к написанию программного обеспечения, рекомендации по его тестированию, описываются необходимые экспериментальные исследования.

4.1 Сроки выполнения курсового проекта и контрольные точки

Курсовой проект выполняется в девятом семестре. Как правило, в течение семестра проходит 9 консультаций. На первой выдается задание. Далее проверяются следующие контрольные точки:

- 3-я консультация - описание выбранного алгоритма;
- 5-я консультация - программная реализация;
- 7-я консультация - отлаженная программа;
- 9-я консультация - зачет.

На зачет студенты предоставляют работающую программу и пояснительную записку, оформленную в соответствии с Приложением 3.

4.2 Требования к программному обеспечению (ПО), создаваемому при выполнении курсового проекта

Программа должна быть реализована в двух частях: головная программа, которая тестирует алгоритм, и подпрограмма, реализующая заданное преобразование. Эти части должны быть размещены в отдельных файлах, так как у преподавателя должна быть возможность как посмотреть результаты работы программы на тестах, предложенных студентом, так и подключить созданную подпрограмму в свой модуль для ее независимого тестирования.

Рекомендуется писать ПО на языке программирования C++. Специальных требований по пользовательскому интерфейсу не предъявляется.

На вход подпрограммы дается квадратный вещественный массив, на выход два массива вещественной и мнимой части комплексного преобразования. Внутри программы данные хранятся в структуре, соответствующей выбранной алгебре. Отметим, что операции сложения и умножения элементов алгебры, а также умножения на комплексные корни целесообразно оформлять отдельными процедурами.

Ниже приводится пример оформления программы (рекомендуемое, но необязательное).

4.2.1 Пример оформления программы

Header-file (например, dft.h)

```
#ifndef _TDFT
#define _TDFT


class TDFT
{
 private:
    // здесь все выделяемые массивы, кроме тех, что идут через параметры, и внутренние процедуры, которые вам нужны

 public:
 TDFT();  // инициализация
 ~TDFT();
 int DFT(float **x_re, float **x_im, int inv, int n);      //комплексное ДПФ
 int Init(int n);    // инициализация всех массивов и выделение памяти
 int Roots(int n);    // подготовка массивов значений корней из единицы
};

#endif
```

Здесь приняты следующие обозначения:

**x_re - входной/выходной двумерный массив, вещественная часть данных,

**x_im - входной/выходной двумерный массив, мнимая часть данных, на входе данных не содержит,

inv - направление преобразования (1 - прямое, -1 - обратное),

n - размер преобразования $n \times n$ (везде выше в тексте использовалось обозначение N.

Под массивы x_re, x_im в вызывающей программе должна быть отведена память размером $n \times n$.

Функции DFT, Init и Roots возвращают 0 при нормальной работе и любое другое число при ошибке.

Файл подпрограммы (например, dft.cpp) и головная программа

```
#include <math.h>
#include <stdlib.h>

#include "DFT.h"

//-------------------------------------------------
//-------------------------------------------------
//-------------------------------------------------

TDFT::TDFT()            // конструктор - обнуление всех переменных и массивов
{
например, так: w=NULL;
}

//-------------------------------------------------

TDFT::~TDFT()           // деструктор - освобождение памяти
{
например, так:    w  = free ( w );
}

//-------------------------------------------------

int TDFT::Init(int n)     // инициализация всех массивов и выделение памяти
{ int n4=n/4, n2=n/2;

распределение памяти под внутренние массивы

  return 0;
}

// Инициализация массивов корней из единицы
int TDFT::Roots(int n)     // инициализация всех массивов и выделение памяти
{
например, как в образце
  return 0;
}
```

```
int TDFT::DFT(float **x_re, float **x_im, int inv, int n)
{
Шаг 1. Перестановка входного массива в порядке р- ичной инверсии

Шаг 2. Передача вещественных данных из массива x_re во внутреннюю структуру данных, согласованную с заданным преобразованием

Шаг 3. Само преобразование. Можете его разбивать на отдельные процедуры.
if (inv==1)
 {
//прямое преобразование
  }//if inv==1
  else
  {
//обратное преобразование
  }//inv=-1

//Переход к комплексному результату в массивы x_re, x_im

return(0);
}
```

В головной (тестирующей) программе последовательность обращений к функциям класса выглядит так.

```
#include "DFT.h"

... main…

…
TFFT1D *pDFT;   //Описание объекта "класс"


pDFT = new TFFT1D; //Выделение под него памяти

pDFT ->Init(n); //Инициализация

pDFT ->Roots(n); //Формирование корней
```

…

Задание размера
Выделение памяти под входные/выходные данные, заполнение массива x_re

pDFT ->DFT(x_re, x_im, inv, n); //само преобразование

Выдача результата, сравнение с тестовым результатом

delete pDFT; //удаление объекта, освобождение памяти

4.3 Рекомендации по тестированию ПО

ПО, разработанное в ходе выполнения курсового проекта, обязательно должно быть оттестировано, так как сложность алгоритмов, предложенных в темах, не позволяет оценить правильность работы программы без специальных тестов. Ниже приводится несколько стандартных способов тестирования алгоритмов дискретных преобразований. Внесение студентами предложений по новым способам тестирования - поощряется.

4.3.1 Пересчет результатов вручную

Одним из первых способов, который следует применить для проверки работы программы - это расчет ожидаемых результатов работы программы для малых объемов входных данных - вручную. Таким способом проверяют правильность перестановки входных данных и расчеты на 1-2 шаге декомпозиции. Очевидно, что входные данные в этом случае должны быть простыми, но не тривиальными. Потому что слишком простой характер входного массива (например, константа) не позволит выявить ряд ошибок.

4.3.2 Использование в качестве входного сигнала базисных функций преобразования

Для предварительного тестирования работы алгоритма на больших длинах в качестве входного сигнала могут быть использованы массивы, соответствующие

значениям базисных функций. В случае двумерного преобразования Фурье базисными функциями являются произведения степеней корней из единицы:

$$h_{k_1k_2}\left(n_1,n_2\right)=\omega_1^{k_1n_1}\omega_2^{k_2n_2}\ ,\ 0\le n_1,n_2,m_1,m_2\le N-1\,, \tag{4.1}$$

где по-прежнему $\omega_1=\exp\left\{2\pi i/N\right\}$, $\omega_2=\exp\left\{2\pi j/N\right\}$ (см. п.2.1).

Подавая такие данные на вход преобразования, в силу ортогональности базисных функций получим

$$\begin{aligned}H\left(m_1,m_2\right)&=\sum_{n_1,n_2=0}^{N-1}\omega_1^{m_1n_1}h_{k_1k_2}\left(n_1,n_2\right)\omega_2^{m_2n_2}=\\&=\sum_{n_1,n_2=0}^{N-1}\omega_1^{m_1n_1}\omega_1^{-k_1n_1}\omega_2^{-k_2n_2}\omega_2^{m_2n_2}=\\&=\sum_{n_1,n_2=0}^{N-1}\omega_1^{\left(m_1-k_1\right)n_1}\omega_2^{\left(m_2-k_2\right)n_2}=\begin{cases}1, & \text{при } m_1=k_1,\,m_2=k_2,\\0, & \text{в остальных случаях.}\end{cases}\end{aligned} \tag{4.2}$$

Однако в предполагаемом ПО на вход преобразования подается вещественный сигнал. Поэтому необходимо заменить функцию (4.1) произведением косинусов или синусов, например, так:

$$\begin{aligned}\tilde{h}_{k_1k_2}\left(n_1,n_2\right)&=\cos\left(2\pi k_1n_1/N\right)\cos\left(2\pi k_2n_2/N\right)=\left(\frac{\omega_1^{k_1n_1}+\omega_1^{-k_1n_1}}{2}\right)\left(\frac{\omega_2^{k_2n_2}+\omega_2^{-k_2n_2}}{2}\right)=\\&=\frac{1}{4}\left(\omega_1^{k_1n_1}\omega_2^{k_2n_2}+\omega_1^{-k_1n_1}\omega_2^{k_2n_2}+\omega_1^{k_1n_1}\omega_2^{-k_2n_2}+\omega_1^{-k_1n_1}\omega_2^{-k_2n_2}\right).\end{aligned} \tag{4.3}$$

При таком входном сигнале, на основании соотношения (4.2) и свойств линейности и периодичности преобразования Фурье, получим на выходе:

$$\tilde{H}(m_1,m_2)=\begin{cases}1/4, & \text{при } (m_1=k_1, m_2=k_2),\\ & \quad (m_1=k_1, m_2=N-k_2),\\ & \quad (m_1=N-k_1, m_2=k_2),\\ & \quad (m_1=N-k_1, m_2=N-k_2),\\ 0, & \text{в остальных случаях.}\end{cases}$$

Входной сигнал легко генерируется программно, а выходной легко контролируется на правильность. Недостатком этого способа является большой объем тестирования, так как для полной уверенности в правильной работе программы необходимо проверить все различные функции вида (4.3). Поэтому рекомендуется проверить программу на нескольких таких тестах и, в случае правильной работы на них, перейти к одному из следующих способов тестирования.

4.3.3 Сравнение с результатами "прямой реализации" преобразования

Этот способ требует написания дополнительного программного модуля, который бы реализовывал преобразование (2.2) впрямую, то есть непосредственным вычислением по формуле

$$X(m_1,m_2)=\sum_{n_1,n_2=0}^{N-1}\omega_1^{m_1n_1}x(n_1,n_2)\omega_2^{m_2n_2}\ .$$

Входной массив $x(n_1,n_2)$, $0\le n_1,n_2\le N-1$ должен содержать случайные числа. Для проверки работы программы необходимо вычислить результаты преобразования как быстрым, так и прямым алгоритмом и сравнить их. Сравнение производить в четырехмерной алгебре, не переходя к комплексному спектру. Результаты обязаны совпасть до нескольких знаков после запятой.

4.3.4 Использование стандартных пакетов программ

Для проверки результатов преобразования после их перевода в комплексную алгебру можно воспользоваться любым стандартным пакетом программ, включающим дискретное преобразование Фурье. Технология аналогична описанной в п.4.2.3.

4.3.5 Проверка учета вещественности входного сигнала

Для проверки этой части программы достаточно запустить ее на одних и тех же входных данных в двух режимах - с учетом вещественности входного сигнала и без нее. Очевидно, что результаты работы должны совпадать. Проверку необходимо проводить в четырехмерной алгебре.

4.4 Экспериментальные исследования, которые необходимо выполнить в ходе проекта

После написания, отладки и тестирования ПО необходимо исследовать быстродействие написанной программы. Для этого ее нужно запускать на входных данных различных размеров $N \times N$ и замерять время выполнения t. Затем построить график зависимости $t(N)$. Далее необходимо таким же образом исследовать что-либо из следующего:

- время вычисления двумерного ДПФ в стандартном пакете,
- время работы прямого вычисления ДПФ,
- время работы быстрого алгоритма без учета вещественности

и также отразить результаты на графике. Сделать выводы.

ПРИЛОЖЕНИЕ 1.
СПИСОК ТЕМ ДЛЯ КУРСОВОГО ПРОЕКТИРОВАНИЯ

1. Реализация и исследование совмещенного алгоритма двумерного вещественного ДПФ на основе ДПФ по основанию 2 с представлением данных в алгебре кватернионов.
2. Реализация и исследование совмещенного алгоритма двумерного вещественного ДПФ на основе ДПФ по основанию 4 с представлением данных в алгебре кватернионов.
3. Реализация и исследование совмещенного алгоритма двумерного вещественного ДПФ на основе ДПФ с расщеплением основания с представлением данных в алгебре кватернионов.
4. Реализация и исследование быстрого алгоритма двумерного вещественного ДПФ по основанию 2 с представлением данных в алгебре кватернионов.

5. Реализация и исследование быстрого алгоритма двумерного вещественного ДПФ по основанию 4 с представлением данных в алгебре кватернионов.
6. Реализация и исследование быстрого алгоритма двумерного вещественного ДПФ с расщеплением основания с представлением данных в алгебре кватернионов.
7. Реализация и исследование совмещенного алгоритма двумерного вещественного ДПФ на основе ДПФ по основанию 2 с представлением данных в гиперкомплексной алгебре.
8. Реализация и исследование совмещенного алгоритма двумерного вещественного ДПФ на основе ДПФ по основанию 4 с представлением данных в гиперкомплексной алгебре.
9. Реализация и исследование совмещенного алгоритма двумерного вещественного ДПФ на основе ДПФ с расщеплением основания с представлением данных в гиперкомплексной алгебре.
10. Реализация и исследование быстрого алгоритма двумерного вещественного ДПФ по основанию 2 с представлением данных в гиперкомплексной алгебре.
11. Реализация и исследование быстрого алгоритма двумерного вещественного ДПФ по основанию 4 с представлением данных в гиперкомплексной алгебре.
12. Реализация и исследование быстрого алгоритма двумерного вещественного ДПФ с расщеплением основания с представлением данных в гиперкомплексной алгебре.
13. Реализация и исследование быстрого алгоритма двумерного вещественного ДПФ по основанию 2 с представлением данных в прямой сумме комплексных алгебр.
14. Реализация и исследование быстрого алгоритма двумерного вещественного ДПФ по основанию 4 с представлением данных в прямой сумме комплексных алгебр.
15. Реализация и исследование быстрого алгоритма двумерного вещественного ДПФ с расщеплением основания в прямой сумме комплексных алгебр.
16. Реализация и исследование быстрого алгоритма двумерного вещественного ДПФ по основанию 3 с представлением данных в кодах Гамильтона-Эйзенштейна.
17. Реализация и исследование быстрого алгоритма двумерного вещественного ДПФ по основанию 6 с представлением данных в кодах Гамильтона-Эйзенштейна.

ПРИЛОЖЕНИЕ 2.
РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Ахмед, Н. Ортогональные преобразования при обработке цифровых сигналов / Н.Ахмед, К.Р.Рао. - М.: Связь, 1980.
2. Блейхут, Н. Быстрые алгоритмы цифровой обработки сигналов / Р.Блейхут. - М.: Мир, 1989
3. Методы компьютерной обработки изображений //под ред. В.А. Сойфера. -М.: Физматлит, 2003.
4. Прэтт, У.К. Цифровая обработка изображений. В 2 т. / У.К.Прэтт. - М.: Мир, 1982.
5. Сороко, Л.М. Спектральные преобразования на ЭВМ / Л.М.Сороко, Т.А.Стриж. - Дубна: ОИЯИ, 1972.
6. Ярославский, Л.П. Введение в цифровую обработку изображений / Л.П.Ярославский. - М.: Советское радио, 1979.
7. Ярославский, Л.П. Цифровая обработка сигналов в оптике и голографии: Введение в цифровую оптику / Л.П.Ярославский. - М.: Радио и связь, 1987
8. Chernov, V.M. Fast algorithms of Discrete Orthogonal Transforms for Data Represented in Cyclotomic Fields / V.M.Chernov // Pattern Recognition and Image Analysis, 1993, V. 3, No. 4. P. 455-458.
9. Алиев, М.В. Многомерное гиперкомплексное ДПФ: параллельный подход / М.В.Алиев, М.А.Чичева //Компьютерная оптика, 2005, №27. С.135-137.
10. Chernov, V. M. Arithmetic methods in the theory of discrete orthogonal transforms / V.M.Chernov // Image Processing and Computer Optics (DIP-94), Proc. SPIE 2363, 1995. P. 134-141.
11. Aliev, M.V. Synthesis of Two-Dimensional DFT Algorithms in the Hypercomplex Algebra / M.V.Aliev //Pattern Recognition and Image Analysis, International Academic Publishing Company “Nauka/Interperiodica, V. 13, No. 1, 2003. P. 61-63.
12. Aliev, M.V. Fast algorithms of d-dimensional DFT of real signal in commutative-associative algebras of 2d dimensionality over the real number field / M.V.Aliev // Computer optics, No. 24, 2002. P. 130-136.
13. Aliev, M.V. Two-dimensional FFT-like algorithms with overlapping / M.V.Aliev, V.M.Chernov // Optical Memory and Neural Networks (Information Optics), V. 11, No. 1, 2002. P. 29-38, 2002.
14. Чичева, М.А. Теоретическая оценка эффективности распараллеливания многомерного гиперкомплексного ДПФ / М.А. Чичева //Тр. науч.-техн. конф. с междунар. участием "Перспективные информационные технологии в научных исследованиях, проектировании и обучении" (ПИТ-2006). - Самара, 2006. Т. 2. С.131-136.
15. Chicheva, M.A. Theoretical and experimental estimation of hypercomplex discrete Fourier transform parallelization efficiency / M.A.Chicheva //Proceedengs of The 2007 International Workshop on Spectral Methods and Multirate Signal Processing, **2007.** P.241-248.

ПРИЛОЖЕНИЕ 3.
ОБРАЗЕЦ ОФОРМЛЕНИЯ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА»

Кафедра геоинформатики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту по дисциплине
"*Компьютерная алгебра*"
Тема № XXX

Студент:	**XXXXXXX**
	группа 657
Руководитель проекта:	***Чичева М. А.***
Оценка:	_____
Дата:	_____

Самара 200**X**

ЗАДАНИЕ

Тема проекта

ОГЛАВЛЕНИЕ

В пояснительную записку необходимо включать только те разделы теории, которые соответствуют заданной теме. Особенно подробно необходимо описать разработанное программное обеспечение, написать руководство пользователя. Необходимо описать тесты, показать какие данные вы подаете на вход, что получаете на выходе, пояснить, почему это верный результат. Раздел "Исследование алгоритма" должен включать таблицы, графики и выводы.

ПРИЛОЖЕНИЕ 4.
АЛГОРИТМЫ ПЕРЕСТАНОВОК
(ДВОИЧНАЯ ИНВЕРСИЯ И АНАЛОГИ)

П4.1. Алгоритм перестановки одномерного массива в порядке двоичной инверсии

Ниже приводится программный код перестановки элементов одномерного массива в порядке двоичной инверсии. В случае двумерного массива данных необходимо переставлять как строки, так и столбцы. Перестановка выполняется "на месте" без дополнительных затрат памяти.

```
dl=n/2; st=n-1;
   j=0;
   for (i=0; i<st; i++)
   { if(i<j) {
                  s0=x[i];
                  x[i]=x[j];
```

```
                x[j]=s0;
            }
        k=dl;
        while (k<=j) {j-=k; k>>=1;}
        j+=k;
        px0_r++; px0_i++;
    }
```

Напомним, что двоично-инверсный порядок индексов получается путем перестановки цифр двоичного кода числа в обратном порядке. Например, для N=8 связь прямого и инверсного порядка индексов показана в таблице П4.1.

Таблица П4.1. Связь прямого и инверсного порядка индексов

Индексы в прямом порядке	Двоичный код	Обратный двоичный код	Индексы в двоично-инверсном порядке
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Такая перестановка применяется в алгоритмах двумерного ДПФ с декомпозицией по основанию 2 и 4.

П4.2. Алгоритм перестановки одномерного массива в порядке троичной инверсии

Такая перестановка используется в алгоритмах двумерного ДПФ с декомпозицией по основанию 3 и является аналогом перестановки, приведенной в предыдущем параграфе. Пример перестановки для N=9 приведен в таблице П4.2. Как и в предыдущем случае, этот алгоритм должен быть применен к строкам и столбцам входной матрицы.

```
dl=n/3;
dl2=dl/2;
st=n-1;
j=0;
for(i=0; i<st; i++)
{ if (i<j) {
```

```
            r1=x[i];
            x[i]=x[j];
            x[j]=r1;
          }
  k=dl; k2=dl2;
  while (k2<=j) {j-=k2; k/=3; k2=k<<1;}
  j+=k;
  px0++;
}
```

Таблица П4.2. Связь прямого и инверсного порядка индексов для N=9

Индексы в прямом порядке	Индексы в троично-инверсном порядке
0	0
1	3
2	6
3	1
4	4
5	7
6	2
7	5
8	8

П4.3. Алгоритм перестановки одномерного массива в порядке инверсии с основанием 6

Такая перестановка используется в алгоритмах двумерного ДПФ с декомпозицией по основанию 6. Как и в предыдущих случаях, этот алгоритм должен быть применен к строкам и столбцам входной матрицы.

```
dl=n/6;
dl2=dl*5;
st=n-1;
j=0;
for(i=0; i<st; i++)
{ if (i<j) {
            r1=x[i];
            x[i]=x[j];
            x[j]=r1;
          }
```

```
  k=dl; k2=dl2;
  while (k2<=j) {j-=k2; k/=6; k2=k*5;}
  j+=k;
  px0++;
}
```

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Дайте определение алгебраических структур, используемых в курсовом проекте.

2. Запишите алгоритм двумерного дискретного преобразования Фурье для произвольного основания декомпозиции p.

3. Выведите вычислительную сложность алгоритмов ДПФ при учете вещественности входного сигнала первым и вторым способом.

4. Сравните сложность своего алгоритма и подобного в другой алгебраической структуре.

5. Сравните сложность своего алгоритма и алгоритма ДПФ в той же алгебре, но другой структуры.

6. Выпишите параллельный алгоритм для декомпозиции по основанию 4.

7. Оцените сложность алгоритмов с расщеплением основания в алгебре кватернионов и алгебре гиперкомплексных чисел.

8. Сравните эффективность двух подходов к учету вещественности входного сигнала.

9. Проанализируйте сложность операций в приведенных алгебрах.

10. Запишите алгоритм перестановки по основанию 10.

ГЛОССАРИЙ

1. Спектр - результат вычисления дискретного преобразования Фурье (ДПФ).

2. Четырехмерная алгебра - алгебра с базисом из четырех элементов.

3. Алгебра кватернионов - некоммутативная четырехмерная алгебра.

4. Гиперкомплексная алгебра - коммутативная четырехмерная алгебра.

5. Коды Гамильтона-Эйзенштейна - способ представления кватернионов.

6. Прямая сумма комплексных алгебр - алгебра изоморфная гиперкомплексной.

7. Гиперкомплексный, кватернионный спектр - результат ДПФ в соответствующей алгебраической структуре.

8. Мультипликативная сложность - количество операций умножения в алгоритме.

9. Аддитивная сложность - количество операций сложения в алгоритме.

10. Распараллеливание - построение параллельного алгоритма.

11. Ускорение - отношение времени последовательного и параллельного алгоритмов.

12. Эффективность - ускорение, отнесенное к числу процессоров.

Учебное издание

Чичева Марина Александровна

КОМПЬЮТЕРНАЯ АЛГЕБРА.
МЕТОДИЧЕСКИЕ УКАЗАНИЯ
К КУРСОВОМУ ПРОЕКТИРОВАНИЮ

Учебное пособие

Научный редактор Э. И. К о л о м и е ц
Редакторская обработка Н. А. Б е р е з и н а
Корректорская обработка Ю. Н. Л и т в и н о в а
Доверстка Ю. Н. Л и т в и н о в а

Подписано в печать 06.11.07. Формат 60x84 1/16.
Бумага офсетная. Печать офсетная.
Печ. л. 4.0.
Тираж 120 экз. Заказ . ИП-109/2007

Самарский государственный
аэрокосмический университет.
443086 Самара, Московское шоссе, 34.

Изд-во Самарского государственного
аэрокосмического университета.
443086 Самара, Московское шоссе, 34.