Национальная Академия наук Украины
Институт проблем регистрации информации

Калиновский Я.А., Ландэ Д.В., Бояринова Ю.Е., Хицко Я.В.

Гиперкомплексные числовые системы и быстрые алгоритмы цифровой обработки информации

Киев – 2013

УДК:004.492

Я.А. Калиновский, Д.В. Ландэ, Ю.Е. Бояринова, Я.В. Хицко.
Гиперкомплексные числовые системы и быстрые алгоритмы цифровой обработки информации. — К.: ИПРИ НАН Украины, 2013. **—** 125 с.

В монографии на основе использования гиперкомплексных числовых систем (ГЧС) рассматриваются вопросы построения алгоритмов быстрого вычисления линейной и циклической сверток, а также моделей информационного поиска. Рассмотрены основы теории ГЧС, обобщены методы построения моделей с помощью ГЧС, проанализированы основные методы вычисления свертки и информационного поиска. Работа сопровождается программным обеспечением выполнения предложенных алгоритмов в среде аналитических вычислений MAPLE.

Для научных работников и специалистов по прикладной математике, которые занимаются проблемами эффективной обработки информации, а также для аспирантов и студентов соответствующих специальностей.

У монографії на основі використання гіперкомплексних числових систем (ГЧС) розглядаються питання побудови алгоритмів швидкого обчислення лінійної та циклічної згорток, а також моделей інформаційного пошуку. Розглянуто основи теорії ГЧС, узагальнені методи побудови моделей за допомогою ГЧС, проаналізовано основні методи обчислення згортки та інформаційного пошуку. Робота супроводжується програмним забезпеченням виконання запропонованих алгоритмів в середовищі аналітичних обчислень MAPLE.

Для науковців і фахівців з прикладної математики, які займаються проблемами ефективної обробки інформації, а також для аспірантів і студентів відповідних спеціальностей.

Рецензенты:
доктор технических наук, профессор А.Я. Матов,
доктор технических наук, профессор В.П. Тарасенко

Редактор Т.В. Синькова

Рекомендовано в печать Ученым советом Института проблем регистрации информации НАН Украины (протокол №13 от 24.12.2013)

ISBN 978-966-02-7127-2

© Я.А. Калиновский
Д.В. Ландэ
Ю.Е. Бояринова
Я.В. Хицко, 2013

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ 6

ВВЕДЕНИЕ 8

1 ОСНОВНЫЕ СВЕДЕНИЯ О ТЕОРИИ ГИПЕРКОМПЛЕКСНЫХ ЧИСЛОВЫХ СИСТЕМ И ИХ ИЗОМОРФИЗМАХ 17

1.1 Основные сведения о теории гиперкомплексных числовых систем 17
1.1.1 Основные определения и операции 17
1.1.2 Нулевой и единичный элементы 18
1.1.3 Некоторые свойства ГЧС 19
1.1.4 Норма гиперкомплексных чисел и их сопряженные 19
1.1.5 Делители нуля и алгоритм деления 22
1.1.6 Экспоненциальная функция от гиперкомплексного переменного 23

1.2 Изоморфные гиперкомплексные числовые системы и методы их построения. 29
1.2.1 Определения и свойства изоморфизма ГЧС 29
1.2.2 Процедуры удвоения Кейли-Диксона и Грассмана-Клиффорда 30
1.2.3 Процедуры умножения размерности и изоморфизм 30
1.2.4 Построение изоморфных пар ГЧС на основе систем двойных чисел W и W_1 31

2 МЕТОДЫ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ МОДЕЛИРОВАНИЯ С ПОМОЩЬЮ ГЧС 37

2.1 Упрощение вида математической модели с помощью ГЧС 37
2.1.1 Уравнения Максвелла 37
2.1.2 Моделирование поворота твердого тела в пространстве с помощью кватернионов 40
2.1.3 Передаточная функция цифрового реверсивного фильтра 41

2.2 Уменьшение объема вычислений с помощью методов ГЧС 43
2.2.1 Применение кватернионов для описания движения твердого тела в пространстве 43
2.2.2 Использование изоморфных переходов 44

3 СЛОЖНОСТЬ ВЫЧИСЛЕНИЙ С ИСПОЛЬЗОВАНИЕМ ИЗОМОРФНЫХ ГЧС 47

3.1 Гиперкомплексное умножение. 47
3.1.1 Ряды изоморфных ГЧС, полученных последовательным удвоением ГЧС W и W_1.. 47
3.1.2 Ряды изоморфных ГЧС на основе системы комплексных чисел C 48

3.2 Расчет норм, сопряженных чисел и нелинейностей 49
3.2.1 Вычисление норм. 50
3.2.2 Вычисление сопряженных чисел 50
3.2.3 Вычисление экспоненциальных функций 51

4 БЫСТРЫЕ АЛГОРИТМЫ ВЫЧИСЛЕНИЯ СВЕРТКИ. 52

4.1 Основные определения. 52
4.1.1 Свертка и корреляция 52
4.1.2 Линейная и циклическая свертки 53
4.2 Основные методы быстрого вычисления свертки. 54
4.2.1 Алгоритм Кука-Тоома. 54
4.2.2 Алгоритм Винограда вычисления коротких сверток. 57
4.2.3 Применение преобразования Фурье. 60
4.2.4 Алгоритм Кули-Тьюки 61
4.2.5 Алгоритм Гуда-Томаса 62
4.3 Линейная свертка с использованием ГЧС. 63
4.4 Декомпозиционный алгоритм вычисления линейной свертки. 72
4.4.1 Метод полной декомпозиции задачи 72
4.4.2 Метод частичной декомпозиции задачи. 73
4.5 Циклическая свертка с использованием ГЧС 75
4.6 Алгоритмы свертки, основанные на циклических сдвигах исходной числовой последовательности 80
4.6.1 Существование двух отсчетов циклической свертки порядка 2^n в гиперкомплексном произведении представлений сворачиваемых последовательностей 80
4.6.2 Циклические сдвиги и точные отсчеты свертки 82
4.6.3 Симметрии оператора изоморфизма систем $W^{(n)}\left(e,2^n\right)$ в $W_1^{(n)}\left(f,2^n\right)$ при циклическом сдвиге преобразуемого числа . 83
4.6.4 Алгоритм циклической свертки с использованием симметрии оператора изоморфизма в случае $2^n=4$ 84
4.6.5 Алгоритм циклической свертки с использованием симметрии оператора изоморфизма в случае $2^n=8$ 86

5 ПРИМЕРЫ МОДЕЛЕЙ НА БАЗЕ ГЧС 90

5.1 Модель информационного поиска с использованием теории гиперкомплексных числовых систем 90
5.2 Гиперкомплексный подход к моделированию репутации в социальных сетях 91
5.3 Исследование моделей 94

ПРИЛОЖЕНИЯ 96

П1 Структура и функции алгоритмически-программной системы вычислений в ГЧС в среде MAPLE 96
П2 Программное обеспечение основных операций с гиперкомплекс-

ными выражениями и числами 97
ПЗ Программное обеспечение быстрых алгоритмов цифровой обработки информации с использованием методов ГЧС 104
БИБЛИОГРАФИЯ 123

ПРЕДИСЛОВИЕ

В Институте проблем регистрации информации Национальной академии наук Украины в течение длительного времени ведутся научные исследования гиперкомплексных числовых систем. Инициатором и руководителем её был профессор М.В. Синьков. Если вначале исследования носили сугубо прикладной характер, обусловленный применением одной из некоммутативных систем гиперкомплексных чисел – кватернионов – к решению задачи ориентации твердого тела в трехмерном пространстве, то в дальнейшем они были расширены в направлении исследований фундаментальных свойств гиперкомплексных числовых систем (ГЧС). Так, были рассмотрены построения модулярных систем в пространствах различных ГЧС, различные аспекты задачи перечисления канонических коммутативных ГЧС, исследованы аналитические свойства функций гиперкомплексного переменного. Изучены законы выполнения операций с гиперкомплексными числами, начиная от элементарных операций – сложения и умножения, и до определения сопряжений, делителей нуля, норм гиперкомплексных чисел в различных системах. Это, в свою очередь, позволило рассмотреть построение различных нелинейностей – от элементарных – типа полиномов и дробно-рациональных функций, до элементарных трансцендентных функций, таких как экспоненциальные и логарифмические функции, а также тригонометрические и гиперболические функции.

Нелинейности последнего типа строятся с помощью разработанного авторами метода ассоциированной с данной ГЧС системы линейных дифференциальных уравнений, который подробно описан в монографии [1]. Он позволяет строить представления элементарных трансцендентных функций в ГЧС любых конечных размерностей как в коммутативных ГЧС, так и в некоммутативных ГЧС, а также и в неканонических ГЧС.

Представления нелинейностей, в частности представления экспоненциальных функций от гиперкомплексного переменного, используются для построения эффективных алгоритмов решения линейных и нелинейных дифференциальных уравнений от гиперкомплексных переменных.

Вышеописанным результатам посвящена монография М.В.Синькова, Я.А. Калиновского и Ю.Е .Бояриновой «Конечные гиперкомплексные числовые системы. Основы теории. Применения» [1], вышедшая в 2010 году.

Дальнейшие исследования авторов походили в направлении изучения изоморфных ГЧС. Здесь были изучены методы генерации изоморфных ГЧС и на их основе разработана обобщенная процедура умножения размерности ГЧС. Она позволяет строить ряды пар изоморфных ГЧС возрастающей размерности с сильно- и слабо-заполненными таблицами умножения, а также дает возможность строить операторы изоморфизма в явном виде.

Этот подход позволяет повысить эффективность как построения алгебраических характеристик ГЧС, так и проведения в них различных операций. Исследованы оценки повышения эффективности вычислений с гиперкомплексными числами при использовании этого принципа изоморфного перехода.

Особое внимание уделяется исследованию вопроса установления изоморфизма между ГЧС. Существующие подходы требуют использования методов решения систем квадратичных уравнений больших размерностей, что часто не дает возможности получения ответа об изоморфности конкретных ГЧС. Авторами предложен и опробован эффективный метод решения, основанный на использовании представлений экспоненциальных функций в изучаемых ГЧС, что значительно снижает трудоемкость решения задачи.

Эти результаты вошли в монографию Я.А. Калиновского и Ю.Е. Бояриновой «Высокоразмерные изоморфные гиперкомплексные числовые системы и их применения» [2], которая вышла в свет в 2012 г.

Теоретические результаты, полученные как авторами, так и другими исследователями, позволили создавать математические модели явлений в различных отраслях науки и техники. Здесь следует отметить, что использование методов гиперкомплексных числовых сис-

тем в математическом моделировании приводит по крайней мере к двум положительным эффектам:

1) упрощению самого вида математической модели;
2) уменьшению объема вычислений при функционировании математической модели.

В данной работе будут рассмотрены некоторые примеры упрощения вида математической модели. Что касается уменьшения количества вычислений, то, как показано в нашей работе [2], основным методом здесь является переход от ГЧС с сильно-заполненной таблицей умножения к изоморфной ей ГЧС со слабо-заполненной таблицей умножения.

Основным содержанием данной работы есть построение математических моделей быстрых алгоритмов цифровой обработки информации именно с помощью таких изоморфных переходов.

Книга состоит из предисловия, пяти глав, заключения, приложений и библиографии.

В гл. 1, которая носит вводный характер, приводятся в кратком виде те теоретические сведения, которые необходимы при дальнейшем изложении материала.

Гл. 2 посвящена анализу некоторых методов повышения эффективности вычислительных алгоритмов с использованием ГЧС.

В гл. 3. изучается эффективность использования изоморфных переходов при выполнении различных линейных и нелинейных операций в ГЧС.

Гл. 4 является центральной в монографии: в ней излагаются вопросы построения методов быстрого вычисления линейной и циклической сверток числовых массивов, основанных на применении изоморфных переходов между ГЧС с сильно- и слабо-заполненными таблицами умножения.

В гл. 5 излагаются возможности применения методов ГЧС к повышению эффективности информационного поиска в среде Интернет.

Монография сопровождается приведенными в приложениях текстами программного обеспечения, предназначенного для решения задач вычисления линейной и циклической сверток. Программное обеспечение выполнено в среде аналитических вычислений MAPLE с использованием алгоритмически-программной системы вычислений в ГЧС.

Авторы выражают благодарность рецензентам книги д.т.н., проф. А.Я.Матову и д.т.н., проф. В.П.Тарасенко за полезные советы, пожелания и высказывания по материалам книги.

Авторы благодарят редактора книги Т.В.Синькову за нелегкий труд по подготовке рукописи к печати.

ВВЕДЕНИЕ

Как правило, одна и та же задача, сформулированная в математическом виде, может быть решена несколькими методами. Как отмечается в [3], вычислительные алгоритмы должны удовлетворять многочисленным и зачастую противоречивым методам. Можно выделить две группы требований. Первая из них связана с адекватностью модели исходной задаче. Вторая группа требований связана с возможностью реализации данной математической модели на ЭВМ, то есть реальные алгоритмы должны быть эффективными, прежде всего, по числу необходимых операций и по требуемому объему памяти ЭВМ. В данной работе речь будет идти только о выполнении второй группы требований.

Один из основных подходов по повышению эффективности реальных алгоритмов можно сформулировать следующим образом. Нужно перейти к таким другим переменным, к другим типам данных или вообще к другим пространствам входных, выходных или внутренних параметров модели, вычислительная сложность решения задачи в которых, включая прямой и обратный переходы, будет меньше исходного варианта.

Далее будут рассмотрены некоторые примеры такого подхода. Рассмотрим, прежде всего, иллюстративный пример для более прозрачного объяснения идеи.

Пусть необходимо решить задачу определения количества корней и их значений для алгебраического уравнения, взятого из работы [4] (№ 3–76):

$$x^3 + 2x - 5\sqrt{3} = 0.$$

Конечно, это уравнение можно решать разными способами. Можно решать его приближенно итерационным методом, выделив области с единственными корнями, что тоже довольно сложно. Как отмечается в [3], не существует каких-то общих регулярных приемов решения задачи о распределении корней произвольной функции. Да и итерационный процесс занимает много времени. Но подстановка $x = y\sqrt{3}$ приводит к уравнению:

$$3y^3 + 2y - 5 = 0.$$

Левая часть легко разлагается на два множителя:

$$(y-1)\left(3y^2 + 3y + 5\right) = 0,$$

откуда имеем все три решения:

$$x_1 = \sqrt{3}; \quad x_{2,3} = \frac{-\sqrt{3} \pm i\sqrt{17}}{2}.$$

Следует отметить, что нет универсальных подходов к выбору структуры перехода от исходных переменных к таким параметрам, что, в конечном итоге, повысилась бы эффективность обработки информации. В этом смысле весьма поучительным является метод символического изображения синусоидальных функций (см., например, [5]). Этот метод получил широкое распространение в электротехнике, радиотехнике и других дисциплинах.

Пусть синусоидальная функция времени t имеет вид:

$$v = V\,sin(\omega t + \varphi), \qquad (0.1)$$

При расчете цепей переменного тока для таких величин необходимо выполнять следующие операции: сложение, умножение и деление на вещественные и комплексные величины, умножение и деление самих синусоидальных функций. При этом результатом должна быть тоже синусоидальная функция, так что потребуются довольно сложные преобразования. Как показывает анализ, выполнение этих операций приводит к громоздким вычислениям.

Для упрощения вычислений американский электротехник Ч.П.Штейнмец предложил символический метод, который заключается в следующем.

Синусоидальные функции (0.1) заменяются их изображениями – комплексными функциями по правилу:

$$v = V\sin(\omega t + \varphi) \Leftrightarrow V\left(\cos(\omega t + \varphi) + i\sin(\omega t + \varphi)\right) = Ve^{i\varphi}e^{i\omega t}. \qquad (0.2)$$

Величина

$$\dot{V} = Ve^{i\varphi} \qquad (0.3)$$

называется комплексной амплитудой.

Тогда производная и интеграл от синусоидальной функции будут иметь вид (0.2):

$$\frac{dv}{dt} \Leftrightarrow i\omega\dot{V}e^{i\omega t}, \quad \int v dt \Leftrightarrow -i\frac{1}{\omega}\dot{V}e^{i\omega t}$$

При таком представлении синусоидальных функций в результате выполнения всех операций, кроме умножения и деления самих синусоидальных функций, будут получаться члены вида $\dot{V}e^{i\omega t}$. Но именно так составляются основные уравнения, моделирующие цепи переменного тока, – законы Ома и Кирхгофа. Поэтому, если все их члены сократить на множитель $e^{i\omega t}$, то получатся чисто линейные уравнения, коэффициенты и неизвестные в которых имеют вид (0.3). Тем самым значительно упрощается как процесс составления уравнений модели, так и процесс их решения.

Таким образом, прямой переход от синусоидальных величин к комплексным амплитудам описывается выражением (0.2), а обратный переход заключается в выделении коэффициента при мнимой компоненте комплексной амплитуды.

Однако переход в другое пространство параметров, давая некоторые преимущества при оперировании с моделью, как правило, не обеспечивает полной адекватности с исходным процессом. Это требует дополнительной корректировки модели. В рассматриваемом примере выше было оговорено, что исключается рассмотрение произведения синусоидальных величин. Хотя формально можно перемножать комплексные амплитуды, однако результат не будет соответствовать физическому смыслу задачи. Действительно, комплексная мощность в цепи переменного тока пропорциональна разности фаз тока и напряжения:

$$\dot{P} = IU\,e^{i(\psi_I - \psi_U)},$$

а перемножение комплексных амплитуд тока и напряжения дает их сумму:

$$\dot{P} = IU\, e^{i(\psi_I + \psi_U)},$$

что неправильно. Для исправления ситуации в теоретических основах электротехники предлагается при использовании метода символического изображения синусоидальных функций определять комплексную мощность умножением комплексной амплитуды тока не на комплексную амплитуду напряжения, а на величину, сопряженную с ней:

$$\dot{P} = \dot{I}\,\overline{\dot{U}} = IU\, e^{i(\psi_I - \psi_U)}.$$

Такая ситуация будет типичной и для других примеров преобразования параметров моделей.

Одним из наиболее мощных средств решения различных задач, и, прежде всего, решения дифференциальных уравнений, как обыкновенных, так и в частных производных, является использование различных интегральных преобразований.

Интегральное преобразование связывает функцию-изображение $F(s)$ с функцией-оригиналом $f(x)$ с помощью интегрального оператора:

$$F(s) = \int_S K(x,\mathrm{s})\, f(x)dx\,.$$

Здесь $F(s)$, $f(x)$ – в общем случае функции многих переменных, а $K(x,s)$ – ядро интегрального оператора, от вида которого зависит тип преобразования.

Интегральные преобразования замечательны тем, что многим соотношениям и операциям над оригиналами $f(x)$ соответствуют более простые в смысле вычислительной сложности соотношения и операции над их изображениями $F(s)$.

Большинство интегральных преобразований являются обратимыми, то есть по известному изображению можно восстановить оригинал, как правило, также интегральным преобразованием:

$$f(x) = \int_S K^{-1}(s,x)F(s)ds\,.$$

Существует большое количество интегральных преобразований, из которых будут кратко охарактеризованы наиболее часто употребляемые в науке и технике. Это преобразования Радона, Лапласа, Карсона-Хевисайда и Фурье. Более подробные сведения об этих и других интегральных преобразованиях можно получить в специальных работах, например [6].

Следует отметить, что для интегральных преобразований существуют их дискретные аналоги, которые позволяют строить алгоритмы и программы для реализации преобразований в различных программных средах на ЭВМ.

Рассмотрим интегральное преобразование Радона и его использование в вычислительной томографии. Для простоты рассмотрим двухмерное преобразование Радона, хотя оно существует и для большего числа измерений.

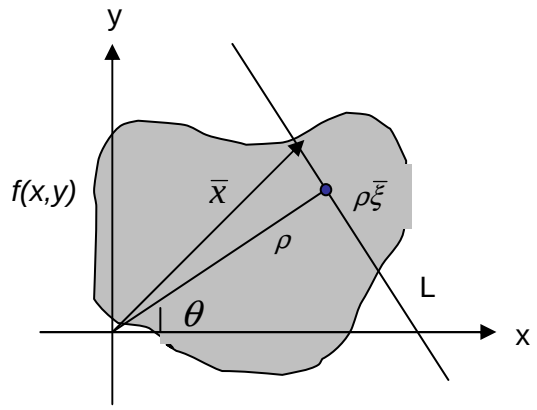


Рис. 0.1 К преобразованию Радона.
(взят с любезного согласия авт. из [10]).

Пусть L – некоторая прямая линия (рис. 0.1) на плоскости (X,Y):

$\rho = x\cos\theta + y\sin\theta$.

$\bar{\xi} = (\cos\theta, \sin\theta)$ – единичный вектор на L.

На плоскости (X,Y) на заштрихованной области задана функция $f(x,y)$, вне которой:

$$f(x,y) = 0.$$

Тогда прямое двухмерное преобразование Радона $\Re$ от $f(x,y)$ – это интегралы двухмерной функции $f(x,y)$ вдоль всевозможных прямых L:

$$\Re f(\rho,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - \rho)dxdy.$$

Как видно из этой формулы, при фиксированных ρ и θ преобразование Радона – это линейный интеграл от функции $f(x,y)$ вдоль прямой L.

Существует и обратное преобразование Радона:

$$\Re^{-1}(\Re f(x,y)) = \frac{1}{4\pi^2}\int_{-\pi}^{\pi}\int_{-\infty}^{\infty}\frac{\partial}{\partial\rho}\Re f(\rho,\theta)\frac{1}{x\cos\theta + y\sin\theta - \rho}d\rho d\theta.$$

Преобразование Радона нашло важнейшее применение в вычислительной томографии, что обусловлено тем, что прямое преобразование Радона выполняется на физическом уровне в самом томографе, где $f(x,y)$ – неизвестная функция, характеризующая плотность изучаемого объекта в сечении по плоскости (X,Y). Обратное преобразование как раз и восстанавливает неизвестную функцию плотности изучаемого объекта в конкретном сечении. На основе формулы обратного преобразования Радона построены многочисленные алгоритмы реконструкции.

Более подробные сведения о преобразовании Радона и методах восстановления можно получить в работах [7–10].

В интегральных преобразованиях Лапласа и Карсона-Хевисайда функция-оригинал заменяется ее изображением – функцией комплексного переменного. Эти преобразования можно применять только к функциям-оригиналам ограниченного роста и удовлетворяющим условиям Дирихле

Кроме того, для упрощения изложения будем считать в дальнейшем, что значения функций-оригиналов и всех их производных до $(n-1)$-й включительно равны нулю при $t = 0$ (нулевые начальные условия). Тогда прямые и обрат-

ные преобразования Лапласа $\Phi[f(t)]$ и Карсона-Хевисайда $F[f(t)]$ имеют соответственно вид:

$$\Phi[f(t)]=\Phi(p)=\int_0^{\infty} e^{-pt} f(t)dt, \quad f(t)=\Phi^{-1}[\Phi(p)]=\frac{1}{2\pi i}\int_{\sigma_1-i\infty}^{\sigma_2+i\infty} e^{pt}\Phi(p)dp$$

$$F[f(t)]=F(p)=p\int_0^{\infty} e^{-pt} f(t)dt, \quad f(t)=F^{-1}[F(p)]=\frac{1}{2\pi i}\int_{\sigma_1-i\infty}^{\sigma_2+i\infty} e^{pt}F(p)dp,\ p\in C.$$

Достоинством преобразования Карсона-Хевисайда является совпадение размерностей изображения и оригинала, что удобно при практических применениях. Однако преобразование Лапласа также нашло применения при решении различных задач.

Из выражений для преобразований Лапласа и Карсона-Хевисайда следует очень простая зависимость между изображением функции-оригинала и ее производными и интегралом при сформулированных выше условиях. А именно, для преобразования Карсона-Хевисайда:

$$F[f'(t)]=pF(p);\ \ F\left[f^{(n)}(t)\right]=p^nF(p);\ \ F\left[\int_0^t f(t)\right]=\frac{1}{p}F(p).$$

Эти преобразования являются линейными операторами, то есть изображение линейной комбинации оригиналов равно линейной комбинации изображений этих оригиналов. Это дает возможность применять преобразования Карсона-Хевисайда (и Лапласа) к системам уравнений, моделирующих, например, электрические цепи. Особенно упрощается при этом задача исследования переходных режимов, так интегро-дифференциальные уравнения переходных режимов преобразуются в алгебраические, решения которых – изображения неизвестных параметров режима. Они будут функциями от p. Для того, чтобы получить функции-оригиналы, нужно сделать обратные преобразования.

Однако, при практических применениях прямые и обратные преобразования выполняют не по формулам, а пользуются заранее составленными таблицами преобразований для многих типов конкретных функций-оригиналов (см, например [5, 6]). Кроме того, существуют методы разложения изображений сложного вида в сумму табличных изображений – т.н. теоремы разложения [6].

Описанные процедуры составляют суть операторного метода решения систем линейных дифференциальных уравнений путем сведения их к системам алгебраических уравнений.

Широкое применение в науке и технике находит преобразование Фурье, которое можно получить из преобразования Лапласа предельным переходом $\mathrm{Re}(p)\to 0$. Формулы прямого и обратного преобразований Фурье имеют вид:

$$\Phi[f(t)]=\Phi(i\omega)=\int_0^{\infty} e^{-i\omega t} f(t)dt, \quad f(t)=\Phi^{-1}[\Phi(i\omega)]=\frac{1}{2\pi}\int_{\infty}^{\infty} e^{i\omega t}\Phi(i\omega)d\omega.$$

Преобразование Фурье обладает всеми свойствами преобразования Лапласа, но оно пригодно для более узкого класса функций. Во всяком случае, функции-оригиналы должны быть абсолютно интегрируемыми.

Прямое преобразование Фурье реализует перевод конкретной задачи из временного пространства в частотное, при котором непериодические функции сводятся к синусоидальным, для которых существуют эффективные методы решения. Таким приемом решаются многие задачи математической физики, построения переходных процессов в линейных электрических цепях и др. Однако нас преобразование Фурье будет интересовать, прежде всего, тем, что для него выполняется теорема свертки: Фурье-изображение свертки равно произведению Фурье-изображений свертываемых функций-оригиналов. Обратным переходом можно найти оригинал свертки. На основе этой теоремы строятся эффективные дискретные алгоритмы свертки числовых последовательностей. Более подробно этот материал рассматривается далее.

Одним из эффективных методов повышения эффективности вычислений является переход от чисел к их изображениям в системах остаточных классов (СОК).

Система остаточных классов – это непозиционная система счисления, числа в которой представляются остатками от деления на выбранную систему оснований [12]. Если задан ряд положительных целых чисел $p_1, p_2, ..., p_n$, называемых основаниями системы, то под системой счисления в остаточных классах будем понимать такую систему, в которой целое положительное число представляется в виде набора остатков (вычетов) по выбранным основаниям $N = (\alpha_1, \alpha_2, ..., \alpha_n)$ причем образование цифр α_i осуществляется следующим процессом:

$$\alpha_i = N - \left[\frac{N}{p_i}\right] p_i \quad \text{для} \quad i = 1,2,..,n,$$

то есть цифра i-го разряда α_i числа N есть наименьший положительный остаток от деления N на p_i. В отличие от обобщенной позиционной системы образование цифры каждого разряда проводится независимо друг от друга. Цифра i-го разряда α_i представляет собой наименьший положительный остаток от деления самого числа N, а не предыдущего частного.

В теории чисел [11] доказано, что если числа p_i взаимно простые между собой, то описанное цифрами представление числа $N = (\alpha_1, \alpha_2, ..., \alpha_n)$ является единственным. Объем диапазона представимых чисел в этом случае равен $\mathrm{P} = p_1 p_2 ... p_n$.

Правила выполнения операций сложения и умножения в системе остаточных классов в случае, если оба числа и результат операции находятся в диапазоне $[0, \mathrm{P})$, а операнды A и B представлены соответственно остатками α_i и β_i по основаниям p_i при $i = 1,2,..., n$:

$$A = (\alpha_1, \alpha_2, ..., \alpha_n),\ B = (\beta_1, \beta_2, ..., \beta_n),$$
$$A \pm B = (\gamma_1, \gamma_2, ..., \gamma_n), \quad \text{где} \quad \gamma_i \equiv \alpha_i \pm \beta_i (mod\ p_i),$$
$$AB = (\delta_1, ..., \delta_n), \quad \text{где} \quad \delta_i \equiv \alpha_i \beta_i (mod\ p_i).$$

Таким образом, выполнение арифметических операций в модулярном коде производится независимо по каждому из модулей, что и указывает на параллелизм данной системы. Это обстоятельство определяет поразрядное выполнение операций. Это свойство позволяет избавиться от необходимости «переносить» единицу старшего разряда, что приводит к появлению кодов с параллельной структурой. Это позволяет распараллелить алгоритмы при выполнении арифметических операций.

Обратный переход от непозиционного представления к обычному позиционному представлению осуществляется по китайской теореме об остатках [12].

Для системы сравнений $x \equiv b_1 (\mathrm{mod}\ p_1)$, $x \equiv b_2 (\mathrm{mod}\ p_2), \ldots,$ $x \equiv b_k (\mathrm{mod}\ p_k)$ с одним неизвестным x, но разными и попарно простыми модулями $(p_i, p_j) = 1$; $i \neq j$, решение можно определить следующим образом: числа M_t, N_t определены из условий $p_1 p_2 \ldots p_k = M_t p_t$, $M_t N_t \equiv 1 (\mathrm{mod}\ p_t)$ и пусть $x_0 = \sum_{i=1}^{k} M_i N_i b_i$. Тогда совокупность значений x, удовлетворяющих системе сравнений, определяется сравнением $x \equiv x_0 (\mathrm{mod}\ p_1 p_2 \ldots p_k)$.

К достоинствам СОК относится: максимальный параллелизм, малоразрядность остатков реализация принципа конвейерной обработки информации, высокая точность, надёжность, способность к самокоррекции. К недостаткам СОК относится: невозможность визуального сопоставления чисел, отсутствие простых признаков выхода результатов операций за пределы диапазона $[0, \mathrm{P})$, ограниченность действия системы сферой целых положительных чисел, получение во всех случаях точного результата операции, что исключает возможность непосредственного приближенного выполнения операций, округления результата и т.п., такие операции, как деление, сравнение и др., требующие информации о величине всего числа, в СОК выполняются по более сложным алгоритмам

При построении быстрых алгоритмов вычисления свертки широко применяется переход к кольцу полиномов. Рассмотрим процесс вычисления линейной свертки двух числовых последовательностей $\bar{a} = \{a_0, a_1, \ldots, a_{n-1}\}$ и $b = \{b_0, b_1, \ldots, b_{l-1}\}$. Их свертка, которую будем обозначать значком $*$, также будет числовой последовательностью, длина которой равна $(n + l - 2)$:

$$\bar{c} = \bar{a} * \bar{b} = \{c_0, c_1, \ldots, c_i, \ldots, c_{n+l-2}\}, \tag{0.4}$$

где:
$$c_i = \sum_{k=0}^{n-1} a_k b_{i-k} . \tag{0.5}$$

Рассмотрим также два полинома $A(x)$ и $B(x)$, степени которых $(n-1)$ и $(l-1)$ соответственно:

$$A(x) = \sum_{i=0}^{n-1} a_i x^i ; \quad B(x) = \sum_{i=0}^{l-1} b_i x^i .$$

Тогда их произведение будет полиномом $(n + l - 2)$-й степени:

$$C(x) = B(x)A(x) = \sum_{k=0}^{n+l-2} a_k b_{i-k} x^k , \qquad (0.6)$$

коэффициенты которого равны отсчетам линейной свертки (0.5). Таким образом, можно перейти от задачи вычисления свертки к задаче умножения полиномов в кольце полиномов. Такой переход целесообразен тем, что кольца полиномов хорошо исследованы, и, как будет показано в дальнейшем, операции в них, особенно умножение, может быть выполнено значительно эффективнее, чем вычисления непосредственно по (0.5). Сами же переходы требуют минимального расхода вычислительных ресурсов.

Одним из методов повышения эффективности алгоритмов математического моделирования является переход к различным гиперкомплексным пространствам.

Комплексные числа нашли широкое применение, как в самой математике, так и в ее многочисленных применениях. С помощью перехода из пространства вещественных чисел в пространство комплексных чисел удобно и компактно формулируются многие математические модели, применяемые в математической физике и в естественных науках – электротехнике, радиотехнике, аэродинамике, гидродинамике, картографии, квантовой механике, теории колебаний, теории хаоса и многих других.

Но система комплексных чисел является одной из систем гиперкомплексных чисел. Поэтому весьма естественным образом возникает вопрос об использовании и других гиперкомплексных числовых систем.

Уже сам создатель первой некоммутативной системы гиперкомплексных чисел – кватернионов, – У.Р.Гамильтон, применял их для решения многочисленных геометрических задач. В дальнейшем кватернионы нашли широкое применение в задачах, связанных с вращением твердого тела в трехмерном пространстве. Также широко применяются в механике дуальные кватернионы – результат применения процедуры удвоения Грассмана-Клиффорда кватернионов с помощью дуальных чисел. В релятивистской физике нашли применение бикватернионы Гамильтона – результат применения процедуры удвоения Грассмана-Клиффорда кватернионов с помощью комплексных чисел. Эти применения обусловлены тем, что через компоненты кватерниона можно выразить кинематические параметры поворота твердого тела. То есть здесь осуществляется переход из пространства кинематических параметров в пространство кватернионов, где вращение моделируется проще.

Если заменить вещественные параметры модели гиперкомплексными параметрами, то модели приобретают какие-то новые качества. Так, например, при замене вещественных коэффициентов передаточных функций фильтров гиперкомплексными значениями, размерность фильтра снижается, и появляется возможность оптимизировать чувствительность фильтра.

В настоящее время не существует каких-то регулярных методов перехода от вещественных параметров моделей в гиперкомплексные области, однако можно сформулировать некоторые частные методы:

– применение таких ГЧС, в которых компоненты гиперкомплексных переменных будут представлять собой параметры моделей или функции от них, а операции с такими гиперчислами будут соответствовать операциям над параметрами моделей в процессе моделирования.

– представление параметров модели в виде гиперкомплексных чисел таким образом, что законы функционирования модели примут достаточно простой вид.

– объединение последовательностей значений параметров модели в одно многомерное гиперкомплексное число таким образом, что обработка последовательности значений сведется к более эффективным операциям над гиперкомплексным числом.

Последний метод характерен для задач обработки цифровых сигналов. Большая часть работы как раз и посвящена исследованиям методов решения задач обработки дискретных сигналов с использованием гиперкомплексных числовых систем.

ГЛАВА 1 ОСНОВНЫЕ СВЕДЕНИЯ О ТЕОРИИ ГИПЕРКОМПЛЕКСНЫХ ЧИСЛОВЫХ СИСТЕМ И ИХ ИЗОМОРФИЗМАХ

1.1 Основные сведения о теории гиперкомплексных числовых систем

1.1.1 Основные определения и операции

Гиперкомплексной числовой системой Γ называется такое кольцо, которое имеет структуру векторного пространства над каким-нибудь полем или кольцом Ω. На аддитивной группе кольца задана операция сложения, а на группоиде кольца – абстрактное умножение. Сложение и умножение связано законами дистрибутивности.

Элементы кольца образуют векторное пространство над каким-нибудь полем или кольцом. Базис векторного пространства образует базис гиперкомплексной числовой системы. Размерность базиса векторного пространства есть размерность ГЧС. Векторное пространство может иметь бесконечное множество базисов, однако у конкретной ГЧС базис зафиксирован. ГЧС с другим базисом – это изоморфная ГЧС. С алгебраической точки зрения эти ГЧС тождественны, но мы их будем различать, так как они разнятся другими характеристиками, важными для нас. Например, количеством вещественных операций при выполнении алгебраических операций с элементами ГЧС.

Рассмотрим ГЧС Γ над Ω. Пусть $a,b\in\Gamma$, $E=\{e_1,e_2,...,e_n\}$ – базис ГЧС, $e_i\in\Gamma$.

Как следует из предыдущего, на Γ заданы три операции:

1) Скалярное умножение – $\alpha a\in\Gamma$. (1.1)
2) Сложение в группе кольца – $a+b\in\Gamma$. (1.2)
3) Умножение в группоиде кольца – $ab\in\Gamma$. (1.3)

Так как Γ – векторное пространство, то каждый его элемент может быть разложен по базисным элементам:

$$a=\sum_{i=1}^{n}a_ie_i\text{, где } a_i\in\Omega. \tag{1.4}$$

Это и будет основным видом гиперкомплексного числа.

Тогда из (1.1) и (1.4) следует:

$$\alpha a=\sum_{i=1}^{n}\alpha a_ie_i\text{,} \tag{1.5}$$

а из (1.2):

$$a+b=\sum_{i=1}^{n}(a_i+b_i)e_i. \tag{1.6}$$

Умножение (1.3) выглядит несколько сложнее:

$$ab=\sum_{i=1}^{n}\sum_{j=1}^{n}a_ib_je_ie_j\text{.}$$

Так как произведение базисных элементов $e_i e_j \in \Gamma$, то оно может быть представлено линейной комбинацией базисных элементов,

$$e_i e_j = \sum_{k=1}^{n} \gamma_{ij}^k e_k \in \Gamma, \quad \gamma_{ij}^k \in \Omega, \quad i, j = 1,...,n, \tag{1.7}$$

коэффициенты которой γ_{ij}^k называются структурными константами ГЧС Γ.

С учетом (1.7) умножение чисел в ГЧС задается формулой:

$$ab = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_i b_j \gamma_{ij}^k e_k. \tag{1.8}$$

Равенства (1.7), а их для одной ГЧС всего n^2, обычно сводят в квадратные двухвходовые таблицы. Например, таблица умножения для системы комплексных чисел C и кватернионов H имеют вид:

C	e_1	e_2
e_1	e_1	e_2
e_2	e_2	e_1

H	e_1	e_2	e_3	e_4
e_1	e_1	e_2	e_3	e_4
e_2	e_2	$-e_1$	e_4	$-e_3$
e_3	e_3	$-e_4$	$-e_1$	e_2
e_4	e_4	e_3	$-e_2$	$-e_1$

Как видно из (1.7), ГЧС полностью и однозначно задается своей размерностью n и набором из n^3 структурных констант γ_{ij}^k.

Равенством

$$e_i e_j = 0 \cup e_k, \quad i, j, k = 1,...,n \tag{1.9}$$

определяется очень важный для приложений класс *канонических* ГЧС.

1.1.2 Нулевой и единичный элементы

В ГЧС вводятся нулевой и единичный элементы. Нулевой элемент – 0– это гиперкомплексное число, нейтральное относительно сложения, а именно:

$$a + 0 = 0 + a = a.$$

Гиперкомплексное число равно нулю, если равны нулю коэффициенты при всех базисных элементах: $a = 0$, если $a_i = 0$, $i = 1,...,n$.

Единичный элемент – это гиперкомплексное число, нейтральное относительно операции умножения. Если обозначить единичный элемент через ε, то должны выполняться такие соотношения:

$$a \cdot \varepsilon = a \cdot \varepsilon = a. \tag{1.10}$$

Единичный элемент может быть элементом базиса системы, как, например, в системах комплексных, двойных, дуальных, квадриплексных чисел, в кватернионах и многих других ГЧС.

Если же единичного элемента в базисе нет, то его можно найти, если решить гиперкомплексное уравнение

$$a \cdot \varepsilon = a \tag{1.11}$$

относительно компонентов единичного элемента, то есть $\varepsilon = \sum_{i=1}^{n} \varepsilon_i e_i$.

1.1.3 Некоторые свойства ГЧС

В зависимости от свойств закона композиции (операции умножения) множество ГЧС разделяется на ряд классов.

Если в системе вещественных чисел умножение имеет свойства коммутативности и ассоциативности, то в ГЧС в общем случае они отсутствуют.

Рассмотрим, прежде всего, свойства коммутативности умножения. Множество ГЧС можно разделить на класс коммутативных ГЧС, для которых выполняется

$$ab = ba, \qquad a, b \in \Gamma$$

и класс некоммутативных ГЧС, для которых

$$ab \neq ba, \qquad a, b \in \Gamma .$$

В классе некоммутативных ГЧС можно выделить класс антикоммутативных ГЧС, для которых

$$ab = -ba, \qquad a, b \in \Gamma .$$

Поскольку произведение двух гиперкомплексных чисел имеет вид (1.9), то признаки коммутативности, некоммутативности и антикоммутативности имеют соответственно вид:

$$\gamma_{ij}^{k} = \gamma_{ji}^{k}, \quad \gamma_{ij}^{k} \neq \gamma_{ji}^{k}, \quad \gamma_{ij}^{k} = -\gamma_{ji}^{k}.$$

Из последнего соотношения вытекает, что базис антикоммутативной ГЧС не может содержать единичный элемент.

Примерами коммутативных ГЧС являются системы комплексных, дуальных, двойных чисел и все их удвоения коммутативными системами.

Рассмотрим свойство ассоциативности умножения. ГЧС ассоциативна, если в ней выполняется тождество

$$(ab)c = a(bc), \qquad a, b, c \in \Gamma .$$

Структурные константы ассоциативной ГЧС должны удовлетворять условиям [1]:

$$\sum_{k=1}^{n} \gamma_{ij}^{k} \gamma_{km}^{l} = \sum_{k=1}^{n} \gamma_{ik}^{l} \gamma_{jm}^{k} .$$

Примерами ассоциативных ГЧС есть системы комплексных, дуальных, двойных чисел и все их удвоения с помощью коммутативных систем. Примером альтернативной ГЧС есть система октав.

1.1.4 Норма гиперкомплексных чисел и их сопряженные

Вопрос о вычислении норм и сопряженных чисел в ГЧС подробно изучается в монографии [1]. Там же приведены формулы для вычисления норм и сопряженных для различных по типу и размерностей ГЧС. Для дальнейшего нам понадобятся только результаты.

Норма гиперкомплексного числа a определяется по формуле

$$N(a)=\left\|\left(N(a)\right)_{j,k}=\sum_{i=1}^{n}\gamma_{ij}^{k}a_i\right\|_{j,k=1,\dots,n}. \qquad (1.12)$$

Как показано в [1], эти нормы обладают свойством мультипликативности. Значительно упрощает определение представления норм следующее утверждение.

Если гиперкомплексная числовая система Γ является прямой суммой нескольких гиперкомплексных числовых систем, то есть $\Gamma=\bigoplus_{i=1}^{n}\Gamma_i$, то ее норма N равна произведению норм N_i всех гиперкомплексных систем, которые входят в прямую сумму:

$$N=\prod_{i=1}^{n}N_i.$$

Докажем верность этого утверждения для случая $n=2$, которое с помощью метода полной математической индукции можно распространить на общий случай. Итак

$$\Gamma=\Gamma_1\oplus\Gamma_2,$$

а размерности систем такие:

$$\dim\Gamma_1=r,\ \ \dim\Gamma_2=s,\ \ \dim\Gamma=r+s.$$

Поскольку матрица нормы определяется соотношением (1.12), то в данном случае она будет иметь вид

$$N(a)=\begin{pmatrix}
\sum_{i=1}^{r}\gamma_{i1}^{1}a_i & \sum_{i=1}^{r}\gamma_{i2}^{1}a_i & \dots & \sum_{i=1}^{r}\gamma_{ir}^{1}a_i & 0 & 0 & \dots & 0\\
\sum_{i=1}^{r}\gamma_{i1}^{2}a_i & \sum_{i=1}^{r}\gamma_{i2}^{2}a_i & \dots & \sum_{i=1}^{r}\gamma_{ir}^{2}a_i & 0 & 0 & \dots & 0\\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots\\
\sum_{i=1}^{r}\gamma_{i1}^{r}a_i & \sum_{i=1}^{r}\gamma_{i2}^{r}a_i & \dots & \sum_{i=1}^{r}\gamma_{ir}^{r}a_i & 0 & 0 & \dots & 0\\
0 & 0 & \dots & 0 & \sum_{i=r+1}^{r+s}\gamma_{i,r+1}^{r+1}a_i & \sum_{i=r+1}^{r+s}\gamma_{i,r+2}^{r+1}a_i & \dots & \sum_{i=r+1}^{r+s}\gamma_{i,r+s}^{r+1}a_i\\
0 & 0 & \dots & 0 & \sum_{i=r+1}^{r+s}\gamma_{i,r+1}^{r+2}a_i & \sum_{i=r+1}^{r+s}\gamma_{i,r+2}^{r+2}a_i & \dots & \sum_{i=r+1}^{r+s}\gamma_{i,r+s}^{r+2}a_i\\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots\\
0 & 0 & \dots & 0 & \sum_{i=r+1}^{r+s}\gamma_{i,r+1}^{r+s}a_i & \sum_{i=r+1}^{r+s}\gamma_{i,r+2}^{r+s}a_i & \dots & \sum_{i=r+1}^{r+s}\gamma_{i,r+s}^{r+s}a_i
\end{pmatrix}$$

По теореме Лапласа

$$\|N(a)\| = \begin{Vmatrix} \sum_{i=1}^{r} \gamma_{i1}^{1} a_i & \sum_{i=1}^{r} \gamma_{i2}^{1} a_i \dots \sum_{i=1}^{r} \gamma_{ir}^{1} a_i \\ \sum_{i=1}^{r} \gamma_{i1}^{2} a_i & \sum_{i=1}^{r} \gamma_{i2}^{2} a_i \dots \sum_{i=1}^{r} \gamma_{ir}^{2} a_i \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ \sum_{i=1}^{r} \gamma_{i1}^{r} a_i & \sum_{i=1}^{r} \gamma_{i2}^{r} a_i \dots \sum_{i=1}^{r} \gamma_{ir}^{r} a_i \end{Vmatrix} \cdot \begin{Vmatrix} \sum_{i=r+1}^{r+s} \gamma_{i,r+1}^{r+1} a_i & \sum_{i=r+1}^{r+s} \gamma_{i,r+2}^{r+1} a_i \dots \sum_{i=r+1}^{r+s} \gamma_{i,r+s}^{r+1} a_i \\ \sum_{i=r+1}^{r+s} \gamma_{i,r+1}^{r+2} a_i & \sum_{i=r+1}^{r+s} \gamma_{i,r+2}^{r+2} a_i \dots \sum_{i=r+1}^{r+s} \gamma_{i,r+s}^{r+2} a_i \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ \sum_{i=r+1}^{r+s} \gamma_{i,r+1}^{r+s} a_i & \sum_{i=r+1}^{r+s} \gamma_{i,r+2}^{r+s} a_i \dots \sum_{i=r+1}^{r+s} \gamma_{i,r+s}^{r+s} a_i \end{Vmatrix} =$$

$$= \|N_1(a_1,\dots,a_r)\| \cdot \|N_2(a_{r+1},\dots,a_{r+s})\|,$$

что и доказывает утверждение.

Из этого утверждения также вытекает мультипликативность нормы прямой суммы Γ, если нормы всех слагаемых Γ_i мультипликативны.

Если в системах размерностью $n=2$ сопряжённое число единственное, как, например, в системе комплексных чисел C, то в ГЧС размерностью $n>2$ их может быть $n-1$ [1]. Но всегда произведение исходного числа и всех $n-1$ сопряжённых чисел должно быть равно норме.

Общий подход поиска сопряженных чисел основывается на использовании выражения для нормы гиперкомплексного числа (1.12). Для сопряженных с a чисел $a_{c1}, \dots, a_{c,\,n\text{-}1}$ должно выполняться равенство

$$a \cdot a_{c1} \cdot \dots \cdot a_{c,n-1} = N(a)\,\varepsilon\,, \qquad (1.13)$$

где ε – единичный элемент рассмотренной ГЧС.

Пусть $$a_{ck} = x_{1k}\,e_1 + x_{2k}\,e_2 + \dots + x_{nk}\,e_n\,, \quad k=1,\dots,n-1. \qquad (1.14)$$

Подставляя (1.14) в (1.13) и приравнивая выражения при одинаковых базисных элементах, получим систему из $(n\text{-}1)$-го алгебраического уравнения от $n(n\text{-}1)$ -го неизвестного. Упрощение этой системы и уменьшение количества неизвестных возможно, если использовать особенности структуры конкретной ГЧС.

Однако, эта система не всегда может иметь решения. Это означает, что из уравнения (1.13) можно определить лишь произведение сопряженных чисел. Но и этого иногда достаточно для решения конкретных задач.

Значительно упрощает определение представления норм следующее. Если гиперкомплексная числовая система Γ является прямой суммой нескольких ГЧС $\Gamma = \bigoplus_{i=1}^{n} \Gamma_i$, и, соответственно, число $a \in \Gamma$ является прямой суммой чисел из систем, которые входят в Γ, то есть $a = \bigoplus_{i=1}^{n} a^{(1)}$, то сопряжённое $\bar{a}$ в гиперкомплексной Γ системе определяется по формуле

$$\bar{a} = \sum_{i=1}^{n} \overline{a^{(i)}} \cdot \prod_{j=1,\,j\neq i}^{n} N_j\left(a^{(j)}\right).$$

Доказательство такое же, как и доказательство подобной формулы для норм.

Выражения для норм и сопряжённых чисел для основных гиперкомплексных числовых систем приведены в работах [1, 2].

1.1.5 Делители нуля и алгоритм деления

Делителем нуля ГЧС является гиперкомплексное число a со свойствами:

1) Оно отлично от нуля: $a \neq 0$.
2) Существует такое отличное от нуля число b из той же гиперкомплексной числовой системы Γ, что произведение чисел a и b равно нулю
$$a \cdot b = b \cdot a = 0; \quad a, b \neq 0; \quad a, b \in \Gamma. \tag{1.15}$$
Таким образом, число b также является делителем нуля.
3) Делитель нуля определяется с точностью до вещественного множителя, то есть, если a – делитель нуля, то и αa – делитель нуля, где $\alpha \in R$.

В соответствии с теоремой Фробениуса [14, 15] только четыре класса изоморфизмов ГЧС не имеют делителей нуля. Это классы, представителями которых являются системы вещественных R, комплексных чисел C, кватернионы H и октавы O.

Как вытекает из (1.15) $N(a \cdot b) = 0$, а, поскольку норма гиперкомплексного числа мультипликативна, то нормы обоих множителей равна нулю. Поскольку оба множителя по определению являются делителями нуля, то без ограничения обобщенности можно определить, что норма делителя нуля равна нулю. Принимая во внимание определение нормы (1.17), получаем уравнение для определения компонентов делителя нуля

$$\left\| \sum_{i=1}^{n} \gamma_{ij}^{k} a_i \right\| = 0. \tag{1.16}$$

Поскольку (1.16) – алгебраическое уравнение n-ой степени от n переменных, то его решение дает n n-мерных решений от $(n-1)$-го параметра. Основной сложностью построения представления делителя нуля является то, что эти решения могут быть комплексными. Но компоненты гиперкомплексного числа должны быть вещественными. Поэтому нужно исследовать, есть ли такие соотношения между переменными, которые превращают коэффициенты мнимых частей в нули, то есть делают решения вещественными.

Как известно [1, 15], в четырех числовых системах – вещественных, комплексных чисел, кватернионов и октав существует деление на любое число этой же системы, кроме нуля. При этом в некоммутативных системах кватернионов и октав существует как левое, так и правое деление, то есть уравнения $ax = b$ и $ya = b$ имеют решения, причем в общем случае они не совпадают: $x \neq y$.

Все эти утверждения полностью переносятся и на ГЧС, которые изоморфны трем из вышеупомянутых числовых систем: комплексным числам, кватернионам и октавам. Напомним, что деление в этих системах сводится к умножению делимого и делителя на число, сопряжённое делителю. Тогда делитель превращается в норму исходного делителя. Норма же числа в этих системах является числом вещественным. Так что деление гиперкомплексных чисел сводится к делению ком-

понентов произведения делимого и сопряжённого к исходному делителю на вещественное число, которое равно его норме.

Докажем тот факт, что $\frac{\overline{a}}{N(a)}$ является обратным числом по отношению к исходному. Действительно, поскольку норма − произведение числа на сопряжённое, то

$$\frac{\overline{a}}{N(a)} = \frac{\overline{a}}{a\overline{a}} = \frac{1}{a} = a^{-1}.$$

Таким образом, можно сказать, что обратное число определяется формулой

$$a^{-1} = \frac{\overline{a}}{N(a)}.$$

На этом основании можно определить деление как произведение делимого на число, обратное делителю

$$\frac{b}{a} = ba^{-1}. \qquad (1.17)$$

1.1.6 Экспоненциальная функция от гиперкомплексного переменного

Так как этот вопрос необходим при изучении изоморфизма гиперкомплексных числовых систем, то рассмотрим его кратко. Полностью он исследуется в работах [1, 2, 16–22].

Пусть рассматривается ГЧС $\Gamma(e,n)$ размерности n с базисом $e=(e_1,...,e_n)$. Во многих случаях экспоненциальную функцию от гиперкомплексного переменного $Exp(X)$ требуется представить в форме гиперкомплексной функции, имеющей вид:

$$F(X) = F\left(\sum_{i=1}^{n} x_i e_i\right) = \left(\sum_{i=1}^{n} f_i(x_1, x_2, ..., x_n)\right) \cdot e_i,$$

где $f_i(x_1, x_2, ..., x_n)$ – вещественные функции от n вещественных переменных.

По аналогии с вещественными и комплексными переменными экспоненциальной функцией $Exp(X)$ от гиперкомплексного переменного X будем считать ее разложение в ряд Маклорена:

$$Exp(X) = \sum_{i=0}^{\infty} \frac{X^i}{i!},$$

где следует считать $X^0 = \varepsilon$.

Построить такое представление непосредственно и, в особенности, для неканонических ГЧС высоких размерностей тяжело. Авторами был разработан обобщенный метод построения, который применим в принципе для любой конечномерной ГЧС. Он заключается в следующем.

Обозначим вектор-столбцы, составленные из компонентов гиперкомплексных переменных, большими латинскими буквами с чертой, например:

$$\overline{X} = (x_1, ..., x_n)^T, . \qquad (1.18)$$

Представление экспоненты в системе $\Gamma(e,n)$ от переменной $M \in \Gamma(e,n)$, которое будем обозначать $Exp(M)$, есть частное решение гиперкомплексного линейного дифференциального уравнения

$$\dot{X} = MX \tag{1.19}$$

при начальном условии

$$Exp(0) = \varepsilon, \tag{1.20}$$

где ε – единичный элемент системы $\Gamma(e,n)$.

Для построения решения гиперкомплексного линейного дифференциального уравнения (1.19) его необходимо представить в векторно-матричной форме. При этом

$$\dot{\overline{X}} = (\dot{x}_1,...,\dot{x}_n)^T, \tag{1.21}$$

а вектор-столбец $\overline{MX}$, полученный из гиперкомплексного числа MX, можно представить в виде матричного произведения некоторой матрицы Ψ размерами $n \times n$, элементы которой есть линейные комбинации компонентов гиперкомплексного числа M, на вектор-столбец $\overline{X}$:

$$\overline{MX} = \Psi\,\overline{X}. \tag{1.22}$$

Тогда гиперкомплексное уравнение (1.19) превратится в систему из n линейных дифференциальных уравнений первого порядка, которая называется ассоциированной системой линейных дифференциальных уравнений:

$$\dot{\overline{X}} = \Psi\,\overline{X}. \tag{1.23}$$

Далее необходимо найти характеристические числа $\lambda_1,...,\lambda_n$ матрицы Ψ, то есть решить характеристическое уравнение

$$\Psi - \lambda E = 0. \tag{1.24}$$

Таким образом, характеристические числа $\lambda_1,...,\lambda_n$ будут зависеть от гиперкомплексного числа M.

Общие решения (1.23), зависят от n произвольных постоянных – $\overline{X}(t,C_1,...,C_n)$. Значения произвольных постоянных устанавливаются с помощью начального условия (1.20). Компоненты вектор-столбца решения $\overline{X}$ и будут компонентами экспоненты от гиперкомплексного числа M:

$$Exp(M) = \sum_{i=1}^{n} \overline{x}_i e_i. \tag{1.25}$$

Метод построения представлений экспоненты от гиперкомплексного переменного с помощью ассоциированной системы линейных дифференциальных уравнений довольно легко формализуется для построения алгоритмов и программ в системах символьных вычислений.

В некоторых задачах целесообразно использовать пары изоморфных ГЧС, одна из которых имеет сильно-заполненную таблицу умножения, а вторая – слабо-заполненную и имеющую диагональный вид. Последнее означает, что на главной диагонали таблицы умножения этой ГЧС размерности $n \geq 1$ стоит m подтаблиц размерности n_m, так что $\sum_{i=1}^{m} n_m = n$, а все остальные элементы таблицы – нули.

Фактически обе эти ГЧС являются прямыми суммами систем вещественных чисел R, комплексных чисел C и неразложимых радикалов.

Изоморфизм двух ГЧС означает существование такого линейного преобразования базисов, детерминант которого не равен нулю, что для операций сложения и умножения образ результата выполнения этих операций равен результату выполнения операции над операндами. Поэтому любое выражение с конечным числом операций преобразуется этим же линейным преобразованием. В таком случае вычисления целесообразно выполнять в той ГЧС, в таблице умножения которой больше нулей. В идеале наибольший эффект дает использование ГЧС диагонального вида, если, конечно, она существует для второй ГЧС пары.

Рассмотрим построение представления экспоненты в одной ГЧС путем использования изоморфного перехода из другой ГЧС, в которой представление экспоненты известно.

Весьма важным обстоятельством является то, что построить представление в ГЧС диагонального типа очень просто. Представление экспоненты от гиперкомплексного числа, которое принадлежит прямой сумме гиперкомплексных числовых систем, равна сумме представлений экспонент от слагаемых этого числа, которые принадлежат всем компонентам прямой суммы.

Однако, необходимо показать, что изоморфное преобразование представления экспоненты от числа в одной ГЧС приведет к представлению экспоненты от образа этого числа в другой ГЧС.

Действительно, пусть даны две изоморфные ГЧС: $\Gamma_1(e,n)$ и $\Gamma_2(f,n)$, и линейное изоморфное преобразование L:

$$\Gamma_1(e,n) \overset{L}{\cong} \Gamma_2(f,n), \tag{1.26}$$

$$L: e_k = \sum_{j=1}^{n} l_{kj} f_j; \quad k = 1,...,n. \tag{1.27}$$

Посмотрим, во что превратится число $X = \sum_{i=1}^{n} x_i e_i \in \Gamma_1(e,n)$ при переходе к системе $\Gamma_2(f,n)$ с помощью изоморфизма L:

$$\begin{aligned} X &= x_1 e_1 + x_2 e_2 + ... + x_n e_n \Leftrightarrow x_1(l_{11} f_1 + l_{12} f_2 + ... + l_{1n} f_n) + \\ &+ x_2(l_{21} f_1 + l_{22} f_2 + ... + l_{2n} f_n) + ... + x_n(l_{n1} f_1 + l_{n2} f_2 + ... + l_{nn} f_n) = \\ &= (x_1 l_{11} + x_2 l_{21} + ... + x_n l_{n1}) f_1 + (x_1 l_{12} + x_2 l_{22} + ... + x_n l_{2n}) f_2 + \\ &+ ... + (x_1 l_{1n} + x_2 l_{2n} + ... + x_n l_{nn}) f_n = y_1 f_1 + ... + y_n f_n \in \Gamma_2(f,n), \end{aligned} \tag{1.28}$$

где

$$y_i = x_1 l_{1i} + x_2 l_{2i} + ... + x_n l_{ni}. \tag{1.29}$$

Тогда

$$\overline{Y} = L^T \overline{X}, \tag{1.30}$$

то есть компоненты гиперкомплексного числа $Y \in \Gamma_2(f,n)$ (вектор-столбец $\overline{Y}$) получаются умножением слева вектор-столбца $\overline{X}$ на транспонированную матрицу оператора изоморфного преобразования L^T.

Значит, если к экспоненте от гиперкомплексного числа X в ГЧС $\Gamma(e,n)$ применить линейное преобразование изоморфизма L, то получится экспонента от гиперкомплексного числа $Y \in \Gamma_2(f,n)$, являющегося образом числа X:

$$Exp(X) = \sum_{i=0}^{\infty} \frac{X^k}{i!} \Leftrightarrow^{L} \sum_{i=0}^{\infty} \frac{Y^i}{k!} = Exp(Y) \in \Gamma_2(f,n). \tag{1.31}$$

Значит, подвергая изоморфному преобразование экспоненту в одной ГЧС, можно получить экспоненту в изоморфной ГЧС от чисел-образов. То же самое можно сказать и о представлениях экспонент, поскольку их построение по степенному ряду даст единственное представление.

Таким образом, если есть две изоморфные, то по представлению экспоненты в одной из ГЧС, подвергнув ее изоморфному преобразованию, можно построить представление экспоненты в другой ГЧС. При этом отпадает необходимость в решении ассоциированной системы линейных дифференциальных уравнений.

Рассмотрим процесс построение представлений экспонент в прямых суммах. Пусть задана гиперкомплексная числовая система Γ размерности n, которая является прямой суммой k числовых систем Γ_i:

$$\Gamma = \bigoplus_{i=1}^{k} \Gamma_i, \tag{1.32}$$

причем, если размерность гиперкомплексной числовой системы Γ_i обозначить d_i, то $\sum_{i=1}^{k} d_i = n$.

Тогда гиперкомплексное число $M \in \Gamma$, можно представить в виде суммы k слагаемых:

$$M = \sum_{i=1}^{n} m_i e_i = \sum_{m=1}^{k} m^{(i)}$$

в соответствии со схемой:

$$m = \underbrace{m_1 e_1 + \ldots + m_{d_1} e_{d_1}}_{m^{(1)} \in \Gamma_1} + \underbrace{m_{d_1+1} e_{d_1+1} + \ldots + m_{d_1+d_2} e_{d_1+d_2}}_{m^{(2)} \in \Gamma_2} + \ldots +$$

$$+ \underbrace{m_{\sum_{i=1}^{s-1} d_i + 1} e_{\sum_{i=1}^{s-1} d_i + 1} + \ldots + m_{\sum_{i=1}^{s} d_i} e_{\sum_{i=1}^{s} d_i} + \ldots + m_{\sum_{i=1}^{k-1} d_i + 1} e_{\sum_{i=1}^{k-1} d_i + 1} + \ldots + m_n e_n}_{m^{(s)} \in \Gamma_s \qquad\qquad m^{(k)} \in \Gamma_k}. \tag{1.33}$$

Так как произведение двух базисных элементов, которые входят в разные слагаемые $m^{(i)}$ представления гиперкомплексного числа (1.33), равно нулю, то все нелинейные операции над этими гиперкомплексными числами сводятся к таким нелинейным действиям отдельно с каждыми слагаемыми в своих гиперкомплексных числовых системах. А это означает, в частности, такое правило возведения числа в целую степень:

$$m^r = \left(\sum_{i=1}^{k} m_i e_i \right)^r = \sum_{m=1}^{k} \left(m^{(i)} \right)^r,$$

и поэтому формула определения экспоненты преобразуется так:

$$Exp(M)=\sum_{i=0}^{\infty}\frac{M^i}{k!}=\sum_{i=1}^{m}\left(\varepsilon_i+\sum_{s=1}^{\infty}\frac{\left(m^{(i)}\right)^s}{s!}\right),$$

или:

$$Exp(M)=\sum_{i=1}^{k}Exp\left(m^{(i)}\right). \tag{1.34}$$

Таким образом, представление экспоненты от гиперкомплексного числа, которое принадлежит прямой сумме гиперкомплексных числовых систем, равно сумме представлений экспонент от слагаемых этого числа, которые принадлежат всем компонентам прямой суммы.

Рассмотрим примеры построения представлений экспонент в гиперкомплексных числовых системах разной размерности по представлениям в изоморфных им системах.

1) Алгебра двойных чисел.

Рассмотрим изоморфные ГЧС W и W_1 с таблицами умножения:

W:

	e_1	e_2
e_1	e_1	e_2
e_2	e_2	e_1

W_1:

	f_1	f_2
f_1	f_1	0
f_2	0	f_2

(1.35)

Изоморфизм между этими ГЧС устанавливается следующими взаимно обратными линейными преобразованиями:

$$\begin{aligned} e_1 &= f_1+f_2; & f_1 &= (e_1+e_2)/2;\\ e_2 &= f_1-f_2; & f_2 &= (e_1-e_2)/2. \end{aligned} \tag{1.36}$$

Посмотрим, как преобразуются числа при переходе от одной системы к другой. Пусть $M=m_1f_1+m_2f_2\in W_1$. Тогда, подставляя вместо базиса f его выражение через базис e, получим:

$$\begin{aligned} M=m_1f_1+m_2f_2 \Leftrightarrow \frac{1}{2}(m_1+m_2)e_1+\frac{1}{2}(m_1-m_2)e_2=\\ =n_1e_1+n_2e_2=N\in W. \end{aligned} \tag{1.37}$$

Из (1.37) имеем выражение компонентов числа $M\in W_1$ через компоненты числа $N\in W$:

$$\begin{aligned} m_1&=n_1+n_2,\\ m_2&=n_1-n_2. \end{aligned} \tag{1.38}$$

Так как $W_1=R\oplus R$, то для $M=m_1f_1+m_2f_2$ в соответствии с сформулированным выше правилом

$$Exp(M)=e^{m_1}f_1+e^{m_2}f_2. \tag{1.39}$$

Используя соотношения (1.36), (1.38) и зависимость между экспоненциальной функцией и гиперболическими функциями

$$e^{\alpha}=ch\alpha+sh\alpha,$$

Получаем цепочку преобразований:

$$Exp(M)=e^{m_1}f_1+e^{m_2}f_2\Leftrightarrow e^{m_1}\frac{1}{2}(e_1+e_2)+e^{m_2}\frac{1}{2}(e_1-e_2)=$$
$$=\frac{1}{2}\left(e^{n_1+n_2}+e^{n_1-n_2}\right)e_1+\frac{1}{2}\left(e^{n_1+n_2}-e^{n_1-n_2}\right)e_2= \qquad (1.40)$$
$$=e^{n_1}\left(ch(n_2)\cdot e_1+sh(n_2)\cdot e_2\right)=Exp(n_1e_1+n_2e_2)$$
$$=Exp(N)\in W.$$

Таким образом, получено представление экспоненты в ГЧС W по ее представлению в системе W_1 (1.44) без построения и решения ассоциированной системы линейных дифференциальных уравнений (1.23):

$$Exp(n_1e_1+n_2e_2)=e^{n_1}\left(ch(n_2)\cdot e_1+sh(n_2)\cdot e_2\right).$$

2) Система триплексных чисел T.

Система триплексных чисел T является неканонической ГЧС. Как известно, она изоморфна ГЧС диагонального типа $R\oplus C$ – прямой сумме систем вещественных R и комплексных чисел C. Таблицы умножения их следующие:

T:

	e_1	e_2	e_3
e_1	e_1	e_2	e_3
e_2	e_2	$(e_3-e_1/2)$	$-e_2$
e_3	e_e	$-e_2$	e_1

$R\oplus C$:

	f_1	f_2	f_3
f_1	f_1	0	0
f_2	0	f_2	f_3
f_3	0	f_3	$-f_1$

Изоморфизм между этими ГЧС устанавливается следующими взаимно обратными линейными преобразованиями:

$$e_1=f_1+f_2;\qquad f_1=\frac{e_1+e_3}{2};$$
$$e_2=f_3;\qquad f_2=\frac{e_1-e_3}{2};$$
$$e_3=f_1-f_2;\qquad f_3=e_2.$$

Пусть

$$N=n_1e_1+n_2e_2+n_3e_3\in T$$
$$M=m_1f_1+m_2f_2+m_3f_3\in R\oplus C.$$

Тогда компоненты чисел преобразуются по правилам [1]:

$$m_1=n_1+n_3;\quad m_2=n_1-n_3;\quad m_3=n_2.$$

Аналогично предыдущему пункту получаем:

$$Exp(M)=e^{m_1}f_1+e^{m_2}(\cos m_3\cdot f_2+\sin m_3\cdot f_3)\Leftrightarrow$$
$$\Leftrightarrow\frac{1}{2}\left(e^{m_1}+e^{m_2}\cos m_3\right)e_1+\frac{1}{2}\left(e^{m_1}-e^{m_2}\cos m_3\right)e_3+e^{m_2}\sin m_3\cdot e_2=$$
$$=\frac{1}{2}\left(e^{n_1+n_3}+e^{n_1-n_3}\cos n_2\right)e_1+e^{n_1-n_3}\sin n_2\cdot e_2+\frac{1}{2}\left(e^{n_1+n_3}-e^{n_1-n_3}\cos n_2\right)e_3.$$

Таким образом, получено представление экспоненты в ГЧС T по ее представлению в системе $R\oplus C$:

$$Exp(M) = Exp(m_1 f_1 + m_2 f_2 + m_3 f_3) =$$
$$= \frac{1}{2}\left(e^{n_1+n_3} + e^{n_1-n_3}\cos n_2\right)e_1 + e^{n_1-n_3}\sin n_2 \cdot e_2 + \frac{1}{2}\left(e^{n_1+n_3} - e^{n_1-n_3}\cos n_2\right)e_3 .$$

1.2 Изоморфные гиперкомплексные числовые системы и методы их построения

1.2.1 Определения и свойства изоморфизма ГЧС

Две гиперкомплексные числовые системы $\Gamma_1(e,n)$ и $\Gamma_2(E,n)$ называются изоморфны (обозначается $\Gamma_1(e,n) \simeq \Gamma_2(E,n)$), если существует такое взаимно однозначное отображение L пространства $\Gamma_1(e,n)$ на пространство $\Gamma_2(E,n)$, что выполняются следующие свойства аддитивности и мультипликативности преобразования:

$$L(A+B) = L(A) + L(B), \tag{1.41}$$
$$L(A\times B) = L(A)\times L(B), \tag{1.42}$$

где $A, B \in \Gamma_1(e,n)$, $L(A), L(B) \in \Gamma_2(E,n)$.

Пусть есть ГЧС $\Gamma_1(e,n)$ и $\Gamma_2(E,n)$ с соответствующими базисами $e = \{e_1, e_2, ..., e_n\}$ и $E = \{E_1, E_2, ..., E_n\}$, между которыми можно установить взаимно однозначное линейное соответствие с вещественной матрицей Λ

$$\begin{aligned} e_1 &= \alpha_{11}E_1 + \alpha_{12}E_2 + ... + \alpha_{1n}E_n ; \\ e_2 &= \alpha_{21}E_1 + \alpha_{22}E_2 + ... + \alpha_{2n}E_n ; \\ &..... \\ e_n &= \alpha_{n1}E_1 + \alpha_{n2}E_2 + ... + \alpha_{nn}E_n , \end{aligned} \tag{1.43}$$

при этом детерминант матрицы Λ отличен от нуля, так как для Λ должно существовать и обратное преобразование Λ^{-1}.

Таким образом, для каждой пары линейно независимых базисов изоморфных гиперкомплексных числовых систем можно найти взаимно однозначное линейное преобразование, которое переводит один базис в другой и наоборот.

Изоморфный переход сохраняет нулевой и единичный элементы систем, то есть нулевой и единичный элементы системы $\Gamma_1(e,n)$ превращаются соответственно в нулевой и единичный элементы системы $\Gamma_2(E,n)$.

Для практики весьма важным является решение вопроса об изоморфности двух заданных ГЧС $\Gamma_1(e,n)$ и $\Gamma_2(E,n)$. Как показано в работе [2], *необходимым и достаточным условием изоморфизма ГЧС* $\Gamma_1(e,n)$ *и* $\Gamma_2(E,n)$ является существование нетривиальных решений системы

$$\sum_{k=1}^{n}\sum_{s=1}^{n}\alpha_{ks}\gamma_{ij}^{k} = \sum_{k=1}^{n}\sum_{s=1}^{n}\sum_{r=1}^{n}\alpha_{ik}\alpha_{js}\gamma_{ks}^{r}; \quad i, j, k \in 1,...,n, \tag{1.44}$$

удовлетворяющих условию $\Lambda^{-1} \neq 0$: если такие решения существуют, то системы $\Gamma_1(e,n)$ и $\Gamma_2(E,n)$ изоморфны, в противном случае – нет. При этом компоненты

решения будут также коэффициентами оператора изоморфизма (1.43).

1.2.2 Процедуры удвоения Кейли-Диксона и Грассмана-Клиффорда

Для получения изоморфных ГЧС можно использовать процедуры удвоения: процедуру Кейли-Диксона и процедуру Грассмана-Клиффорда.[40–45].

Процедуры Кейли-Диксона позволяют получать нормированные ГЧС размерности 2^n, где $n \in N$ — порядок удвоения; причем при $n > 4$ получаемые ГЧС неассоциативные.

Также они позволяют получить более широкие классы изоморфных ГЧС, как по размерности, так и по свойствам.

Процедура удвоения фактически означает то, что компоненты гиперкомплексного числа для данной ГЧС уже не являются вещественными числами, а числами, принадлежащими к какой-либо ГЧС размерности 2. В общем случае, если размерность системы не только равна 2, тогда размерность полученной ГЧС будет уже не удваиваться по отношению к исходной, а умножаться на размерность той ГЧС, элементами которой будут компоненты исходной ГЧС. Такие процедуры будем называть в отличие от процедур удвоения процедурами умножения размерности.

Будем обозначать результат применения процедуры умножения размерности системы $\Gamma_1(e,n)$ системой $\Gamma_2(f,m)$ так: $Д(\Gamma_1(e,n),\Gamma_2(f,m))$.

1.2.3 Процедуры умножения размерности и изоморфизм

При использовании процедуры умножения размерности базис получаемых ГЧС состоит из парных произведений базисных элементов исходных ГЧС. Так как первоначальные ГЧС являлись коммутативными, то и базисные элементы результирующей ГЧС будут коммутативны. Следует отметить, что независимо от порядка расположения ГЧС в операторе умножения размерности, полученные базисы будут одинаковыми. Следовательно, будут идентичными и таблицы умножения ГЧС, полученных при перестановке их в операторе умножения размерности.

Поэтому могут быть сформулированы теоремы о процедуре умножения размерности [2]:

Теорема 1

Если:

1) $\Gamma_1(e,n)$ и $\Gamma_2(f,m)$ коммутативны;

2) $e_i f_j = f_j e_i$, $\forall i = 1,...,n$; $\forall j = 1,...,m$,

тогда при применении процедуры умножения размерности системы можно коммутировать, то есть:

$$Д(\Gamma_1(e,n),\Gamma_2(f,m)) \simeq Д(\Gamma_2(f,m),\Gamma_1(e,n)), \qquad (1.45)$$

причем изоморфизм устанавливается перестановкой строк и столбцов таблиц умножения.

Теорема 2

Пусть $\Gamma_2(f,m) \simeq \Gamma_3(g,m)$,

тогда $Д(\Gamma_1(e,n),\Gamma_2(f,m)) \simeq Д(\Gamma_1(e,n),\Gamma_3(g,m))$, то есть умножение размерности одной и той же ГЧС изоморфными ГЧС приводит к изоморфным ГЧС.

Теорема 3

Пусть $\Gamma_1(e,n) \simeq \Gamma_3(f,n)$ и $\Gamma_2(g,m) \simeq \Gamma_4(h,m)$, $e = L_1 g$, $f = L_2 h$,

тогда $Д(\Gamma_1(e,n),\Gamma_2(f,m)) = \Gamma_5(ef,mn) \simeq \Gamma_6(gh,mn) = Д(\Gamma_3(g,n),\Gamma_4(h,m))$

и существует невырожденное линейное преобразование, переводящее базис $\{ef\}$ в базис $\{gh\}$.

Если же к одной и той же ГЧС применяются процедуры умножения размерности различными ГЧС, но размерности которых одинаковы, то в результате получаются различные ГЧС одинаковой размерности, но, в общем случае, неизоморфные между собой. Если для умножения размерности применяются изоморфные ГЧС, то и результаты после применения процедуры умножения также будут изоморфными между собой.

1.2.4 Построение изоморфных пар ГЧС на основе систем двойных чисел W и W_1

Оператор изоморфизма перехода от гиперкомплексных чисел $W(e,2)$ к системе, состоящей из прямой суммы двух вещественных систем $R \oplus R = W_1(f,2)$ имеет вид [1, 2]:

$$L: \begin{cases} e_1 = f_1 + f \\ e_2 = f_1 - f_2 \end{cases}, \tag{1.46}$$

При этом для умножения двух чисел в системе $W(e,2)$ необходимо выполнить 4 вещественных умножения и 2 сложения, тогда как при переходе к изоморфной системе $R \oplus R = W_1(f,2)$ – всего 2 умножения. Таким образом, можно значительно сократить количество вещественных операций.

Построение изоморфных пар происходит следующим образом.

Пусть:

$$\Gamma_1(e,2) = W(e,2) \simeq W_1(f,2) = \Gamma_3(f,2)$$

и

$$\Gamma_2(g,2) = W(g,2) \simeq W_1(h,2) = \Gamma_4(h,2).$$

Между ГЧС $\Gamma_1(e,2)$ и $\Gamma_3(f,2)$ существуют изоморфные преобразования вида (1.46), а между $\Gamma_2(g,2)$ и $\Gamma_4(h,2)$ – преобразования следующего вида:

$$L_2: \begin{Bmatrix} g_1 = h_1 + h_2 \\ g_2 = h_1 - h_2 \end{Bmatrix}. \tag{1.47}$$

Применяя процедуру умножения размерности к системам Γ_1 и Γ_2:

$$Д(\Gamma_1(e,2),\Gamma_2(g,2)) = \Gamma_5(eg,4)$$

и к системам Γ_3 и Γ_4:

$$Д(\Gamma_3(f,2),\Gamma_4(h,2))=\Gamma_6(fh,4),$$

определяем их таблицы умножения.

Для системы $\Gamma_5(eg,4)$

	e_1g_1	e_1g_2	e_2g_1	e_2g_2
e_1g_1	$e_1e_1\,\Gamma_2$		$e_1e_2\,\Gamma_2$	
e_1g_2				
e_2g_1	$e_1e_2\,\Gamma_2$		$e_2e_2\,\Gamma_2$	
e_2g_2				

$\Rightarrow$

	e_1g_1	e_1g_2	e_2g_1	e_2g_2
e_1g_1	e_1g_1	e_1g_2	e_2g_1	e_2g_2
e_1g_2	e_1g_2	e_1g_1	e_2g_2	e_2g_1
e_2g_1	e_2g_1	e_2g_2	e_1g_1	e_1g_2
e_2g_2	e_2g_2	e_2g_1	e_1g_2	e_1g_1

Для системы $\Gamma_6(fh,4)$

	f_1h_1	f_1h_2	f_2h_1	f_2h_2
e_1g_1	$f_1f_1\,\Gamma_2$		$f_1f_2\,\Gamma_2$	
e_1g_2				
e_2g_1	$f_1f_2\,\Gamma_2$		$f_2f_2\,\Gamma_2$	
e_2g_2				

$\Rightarrow$

	f_1h_1	f_1h_2	f_2h_1	f_2h_2
f_1h_1	f_1h_1	0	0	0
f_1h_2	0	f_1h_2	0	0
f_2h_1	0	0	f_2h_1	0
f_2h_2	0	0	0	f_2h_2

При этом система $\Gamma_5(eg,4)$ — сильно-заполненная, а система $\Gamma_6(fh,4)$ — слабо-заполненная. Линейное преобразование, связывающее базисы $\{eg\}$ и $\{fh\}$:

$$\begin{aligned}
e_1g_1&=(f_1+f_2)(h_1+h_2)=f_1h_1+f_1h_2+f_2h_1+f_2h_2;\\
e_1g_2&=(f_1+f_2)(h_1-h_2)=f_1h_1-f_1h_2+f_2h_1-f_2h_2;\\
e_2g_1&=(f_1-f_2)(h_1+h_2)=f_1h_1+f_1h_2-f_2h_1-f_2h_2;\\
e_2g_2&=(f_1-f_2)(h_1-h_2)=f_1h_1-f_1h_2-f_2h_1+f_2h_2.
\end{aligned}\qquad (1.48)$$

Так как обратное изоморфное преобразование между системами $W(e,2)$ и $R\oplus R=W_1(f,2)$ имеет вид

$$L_1^{-1}:\begin{cases}f_1=(e_1+e_2)/2\\ f_2=(e_1-e_2)/2\end{cases},$$

а обратное преобразование между системами $W(g,2)$ и $W_1(h,2)$:

$$L_1^{-1}:\begin{cases}h_1=(g_1+g_2)/2\\ h_2=(g_1-g_2)/2\end{cases}.$$

Тогда обратное преобразование между системами $\Gamma_5(eg,4)$ и $\Gamma_6(fh,4)$: имеет вид:

$$
\begin{aligned}
f_1 h_1 &= \frac{1}{4}(e_1+e_2)(g_1+g_2) = \frac{1}{4}(e_1g_1+e_1g_2+e_2g_1+e_2g_2);\\
f_1 h_2 &= \frac{1}{4}(e_1+e_2)(g_1-g_2) = \frac{1}{4}(e_1g_1-e_1g_2+e_2g_1-e_2g_2);\\
f_2 h_1 &= \frac{1}{4}(e_1-e_2)(g_1+g_2) = \frac{1}{4}(e_1g_1+e_1g_2-e_2g_1-e_2g_2);\\
f_2 h_2 &= \frac{1}{4}(e_1-e_2)(g_1-g_2) = \frac{1}{4}(e_1g_1-e_1g_2-e_2g_1+e_2g_2).
\end{aligned}
\qquad (1.49)
$$

Следует отметить, что для умножения двух чисел в системе $\Gamma_5(eg,4)$ необходимо 16 вещественных умножений и 12 сложений, а в системе $\Gamma_6(fh,4)$ для этого же действия необходимо 4 вещественных умножения.

Для дальнейшего построения соответствующих изоморфных сильно-заполненных и слабо-заполненных систем вводятся следующие обозначения: $\Gamma_5(eg,4)$ – $W^{(2)}(m,4)$; $\Gamma_6(fh,4)$ – $W_1^{(2)}(M,4)$, показатель в верхних скобках – порядок удвоения исходных систем W и W_1, m,M – обозначение базисов соответствующих систем.

При этом: $W^{(2)}(m,4) \simeq W_1^{(2)}(M,4)$.

Используя эти обозначения, получаются соответствующие таблицы умножения:

$W^{(2)}(m,4)$	m_1	m_2	m_3	m_4
m_1	m_1	m_2	m_3	m_4
m_2	m_2	m_1	m_4	m_3
m_3	m_3	m_4	m_1	m_2
m_4	m_4	m_3	m_2	m_1

$W_1^{(2)}(M,4)$	M_1	M_2	M_3	M_4
M_1	M_1	0	0	0
M_2	0	M_2	0	0
M_3	0	0	M_3	0
M_4	0	0	0	M_4

Тогда прямое изоморфное преобразовании (1.48) и обратное (1.49) перепишутся соответственно в виде:

$$
\begin{aligned}
m_1 &= M_1+M_2+M_3+M_4;\\
m_2 &= M_1-M_2+M_3-M_4;\\
m_3 &= M_1+M_2-M_3-M_4;\\
m_4 &= M_1-M_2-M_3+M_4.
\end{aligned}
\qquad (1.50)
$$

$$
\begin{aligned}
M_1 &= \frac{1}{4}(m_1+m_2+m_3+m_4);\\
M_2 &= \frac{1}{4}(m_1-m_2+m_3-m_4);\\
M_3 &= \frac{1}{4}(m_1+m_2-m_3-m_4);\\
M_4 &= \frac{1}{4}(m_1-m_2-m_3+m_4).
\end{aligned}
\qquad (1.51)
$$

Продолжая процесс умножения размерности, получаем такие ГЧС:

$$Д\left(W^{(2)}(m,4),W(e,2)\right) = W^{(3)}(em,8) \quad \text{и} \quad Д\left(W_1^{(2)}(M,4),W_1(f,2)\right) = W_1^{(3)}(fM,8).$$

Выполняя аналогичные действия, получим таблицу умножения для системы $W^{(3)}(em,8)$:

	e_1m_1	e_1m_2	e_1m_3	e_1m_4	e_2m_1	e_2m_2	e_2m_3	e_2m_4
e_1m_1	e_1m_1	e_1m_2	e_1m_3	e_1m_4	e_2m_1	e_2m_2	e_2m_3	e_2m_4
e_1m_2	e_1m_2	e_1m_1	e_1m_4	e_1m_3	e_2m_2	e_2m_1	e_2m_4	e_2m_3
e_1m_3	e_1m_3	e_1m_4	e_1m_1	e_2m_2	e_2m_3	e_2m_4	e_2m_1	e_2m_2
e_1m_4	e_1m_4	e_1m_3	e_1m_2	e_1m_1	e_2m_4	e_2m_3	e_2m_2	e_2m_1
e_2m_1	e_2m_1	e_2m_2	e_2m_3	e_2m_4	e_1m_1	e_2m_2	e_1m_3	e_1m_4
e_2m_2	e_2m_2	e_2m_1	e_2m_4	e_2m_3	e_1m_2	e_1m_1	e_1m_4	e_1m_3
e_2m_3	e_2m_3	e_2m_4	e_2m_1	e_2m_2	e_1m_3	e_1m_4	e_1m_1	e_2m_2
e_2m_4	e_2m_4	e_2m_3	e_2m_2	e_2m_1	e_1m_4	e_1m_3	e_1m_2	e_1m_1

Таблица умножения для системы $W_1^{(3)}(fM,8)$ имеет вид

	f_1M_1	f_1M_2	f_1M_3	f_1M_4	f_2M_1	f_2M_2	f_2M_3	f_2M_4
f_1M_1	f_1M_1	0	0	0	0	0	0	0
f_1M_2	0	f_1M_2	0	0	0	0	0	0
f_1M_3	0	0	f_1M_4	0	0	0	0	0
f_1M_4	0	0	0	f_1M_4	0	0	0	0
f_2M_1	0	0	0	0	f_2M_1	0	0	0
f_2M_2	0	0	0	0	0	f_2M_2	0	0
f_2M_3	0	0	0	0	0	0	f_2M_3	0
f_2M_4	0	0	0	0	0	0	0	f_2M_4

После переименования элементов базиса по правилам $e_im_j \to e_{4(i-1)+j}$ и $f_iM_j \to M_{4(i-1)+j}$ получим таблицы систем $W^{(3)}(e,8)$ и $W_1^{(3)}(f,8)$ соответственно:

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
e_1	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
e_2	e_2	e_1	e_4	e_3	e_6	e_5	e_8	e_7
e_3	e_3	e_4	e_1	e_2	e_7	e_8	e_5	e_6
e_4	e_4	e_3	e_2	e_1	e_8	e_7	e_6	e_5
e_5	e_5	e_6	e_7	e_8	e_1	e_2	e_3	e_4
e_6	e_6	e_5	e_8	e_7	e_2	e_1	e_4	e_3
e_7	e_7	e_8	e_5	e_6	e_3	e_4	e_1	e_2
e_8	e_8	e_7	e_6	e_5	e_4	e_3	e_2	e_1

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	f_1	0	0	0	0	0	0	0
f_2	0	f_2	0	0	0	0	0	0
f_3	0	0	f_3	0	0	0	0	0
f_4	0	0	0	f_4	0	0	0	0
f_5	0	0	0	0	f_5	0	0	0
f_6	0	0	0	0	0	f_6	0	0
f_7	0	0	0	0	0	0	f_7	0
f_8	0	0	0	0	0	0	0	f_8

Как видим, таблицу системы $W^{(3)}(e,8)$ можно разбить на 4 квадранта. При этом квадранты 1 и 3, а также 2 и 4 соответственно одинаковые.

Продолжим процесс умножения размерности. Рассмотрим процедуры:

$$Д\left(W^{(3)}(m,8),W(e,2)\right)=W^{(4)}(em,16) \text{ и } Д\left(W_1^{(3)}(M,8),W_1(f,2)\right)=W_1^{(4)}(fM,16).$$

Выполняя аналогичные действия, получим таблицы умножения для систем $W^{(4)}(em,16)$ и $W_1^{(4)}(fM,16)$. После переименования элементов базиса по правилам $e_i m_j \to e_{8(i-1)+j}$ и $f_i M_j \to M_{8(i-1)+j}$ получим таблицы систем $W^{(4)}(e,16)$ и $W_1^{(4)}(f,16)$ соответственно:

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}	e_{16}
e_1	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}	e_{16}
e_2	e_2	e_1	e_4	e_3	e_6	e_5	e_8	e_7	e_{10}	e_9	e_{12}	e_{11}	e_{14}	e_{13}	e_{16}	e_{15}
e_3	e_3	e_4	e_1	e_2	e_7	e_8	e_5	e_6	e_{11}	e_{12}	e_9	e_{10}	e_{15}	e_{16}	e_{13}	e_{14}
e_4	e_4	e_3	e_2	e_1	e_8	e_7	e_6	e_5	e_{12}	e_{11}	e_{10}	e_9	e_{16}	e_{15}	e_{14}	e_{13}
e_5	e_5	e_6	e_7	e_8	e_1	e_2	e_3	e_4	e_{13}	e_{14}	e_{15}	e_{16}	e_9	e_{10}	e_{11}	e_{12}
e_6	e_6	e_5	e_8	e_7	e_2	e_1	e_4	e_3	e_{14}	e_{13}	e_{16}	e_{15}	e_{10}	e_9	e_{12}	e_{11}
e_7	e_7	e_8	e_5	e_6	e_3	e_4	e_1	e_2	e_{15}	e_{16}	e_{13}	e_{14}	e_{11}	e_{12}	e_9	e_{10}
e_8	e_8	e_7	e_6	e_5	e_4	e_3	e_2	e_1	e_{16}	e_{15}	e_{14}	e_{13}	e_{12}	e_{11}	e_{10}	e_9
e_9	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}	e_{16}	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
e_{10}	e_{10}	e_9	e_{12}	e_{11}	e_{14}	e_{13}	e_{16}	e_{15}	e_2	e_1	e_4	e_3	e_6	e_5	e_8	e_7
e_{11}	e_{11}	e_{12}	e_9	e_{10}	e_{15}	e_{16}	e_{13}	e_{14}	e_3	e_4	e_1	e_2	e_7	e_8	e_5	e_6
e_{12}	e_{12}	e_{11}	e_{10}	e_9	e_{16}	e_{15}	e_{14}	e_{13}	e_4	e_3	e_2	e_1	e_8	e_7	e_6	e_5
e_{13}	e_{13}	e_{14}	e_{15}	e_{16}	e_9	e_{10}	e_{11}	e_{12}	e_5	e_6	e_7	e_8	e_1	e_2	e_3	e_4
e_{14}	e_{14}	e_{13}	e_{16}	e_{15}	e_{10}	e_9	e_{12}	e_{11}	e_6	e_5	e_8	e_7	e_2	e_1	e_4	e_3
e_{15}	e_{15}	e_{16}	e_{13}	e_{14}	e_{11}	e_{12}	e_9	e_{10}	e_7	e_8	e_5	e_6	e_3	e_4	e_1	e_2
e_{16}	e_{16}	e_{15}	e_{14}	e_{13}	e_{12}	e_{11}	e_{10}	e_9	e_8	e_7	e_6	e_5	e_4	e_3	e_2	e_1

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
f_1	f_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_2	0	f_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_3	0	0	f_3	0	0	0	0	0	0	0	0	0	0	0	0	0
f_4	0	0	0	f_4	0	0	0	0	0	0	0	0	0	0	0	0
f_5	0	0	0	0	f_5	0	0	0	0	0	0	0	0	0	0	0
f_6	0	0	0	0	0	f_6	0	0	0	0	0	0	0	0	0	0
f_7	0	0	0	0	0	0	f_7	0	0	0	0	0	0	0	0	0
f_8	0	0	0	0	0	0	0	f_8	0	0	0	0	0	0	0	0
f_9	0	0	0	0	0	0	0	0	f_9	0	0	0	0	0	0	0
f_{10}	0	0	0	0	0	0	0	0	0	f_{10}	0	0	0	0	0	0
f_{11}	0	0	0	0	0	0	0	0	0	0	f_{11}	0	0	0	0	0
f_{12}	0	0	0	0	0	0	0	0	0	0	0	f_{12}	0	0	0	0
f_{13}	0	0	0	0	0	0	0	0	0	0	0	0	f_{13}	0	0	0
f_{14}	0	0	0	0	0	0	0	0	0	0	0	0	0	f_{14}	0	0
f_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	f_{15}	0
f_{16}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	f_{16}

Таким образом, для умножения двух чисел в системе вида $W_1^{(4)}(f,16)$ необходимо 16 умножений, а в системе $W^{(4)}(e,16)$ необходимо 256 умножений и 12 сложений.

В общем случае, для умножения двух чисел в системах вида $W_1^{(k)}\left(f,2^k\right)$ необходимо k умножений, а в системе $W^{(k)}\left(em,2^k\right)$ необходимо k^2 умножений и k^2-k сложений.

ГЛАВА 2 МЕТОДЫ ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ МОДЕЛИРОВАНИЯ С ПОМОЩЬЮ ГЧС

Как было показано в предыдущем разделе, применение гиперкомплексных числовых систем в математическом моделировании приводит, по крайней мере, к двум положительным результатам. Это, во-первых, придает модели компактный, удобный для обозрения вид, и, во-вторых, может привести к значительному уменьшению количества вычислений, что даже гораздо важнее. В данной главе будут приведены примеры, иллюстрирующие эти утверждения, и будут показаны те методы, с помощью которых получен положительный эффект использования ГЧС. Эти методы в дальнейшем могут быть использованы для расширения круга задач, решаемых с помощью ГЧС.

2.1 Упрощение вида математической модели с помощь ГЧС

2.1.1 Уравнения Максвелла

Уравнения Максвелла представляют собой систему из четырех линейных дифференциальных уравнений в частных производных, выражающих пространственно-временную зависимость между напряженностями электрического E и магнитного H полей, электрической D и магнитной B индукциями, электрическими зарядами ρ и токами j. Все эти величины, кроме ρ, являются трехмерными векторами.

В дифференциальной форме в векторной записи в системе физических единиц $СГС$ уравнения Максвелла имеют вид [см., например, 23]:

$$\begin{aligned} divD &= 4\pi\rho, \\ divB &= 0, \\ rotE &= -\frac{1}{c}\frac{\partial B}{\partial t}, \\ rotH &= \frac{4\pi}{c} j + \frac{1}{c}\frac{\partial D}{\partial t} \end{aligned} \qquad (2.1)$$

или в виде волновых уравнений через скалярный Φ и векторный A потенциалы электромагнитного поля:

$$\begin{aligned} div\, A + \frac{1}{c}\frac{\partial \Phi}{\partial t} &= 0, \\ \nabla^2 \Phi - \frac{1}{c^2}\frac{\partial^2 \Phi}{\partial t^2} &= -4\pi\rho, \\ \nabla^2 A - \frac{1}{c^2}\frac{\partial^2 A}{\partial t^2} &= -\frac{1}{c} 4\pi j. \end{aligned} \qquad (2.2)$$

В работе [24] предлагается упростить эту модель с помощью одной из 4-мерных гиперкомплексных числовых систем – системы квадриплексных чисел K, таблица умножения базисных элементов которой имеет вид:

K	e_1	e_2	e_3	e_4
e_1	e_1	e_2	e_3	e_4
e_2	e_2	$-e_1$	e_4	$-e_3$
e_3	e_3	e_4	$-e_1$	$-e_2$
e_4	e_4	$-e_3$	$-e_2$	e_1

(2.3)

Введем следующие обозначения:

E, J – 4-мерные вектора с компонентами (e_1, e_2, e_3, e_4) и (ρ, j_2, j_3, j_4) соответственно;

$Z = cte_1 + xe_2 + ye_3 + ze_4$ – 4-мерная независимая переменная;

$\Psi = \Phi e_1 + A_x e_2 + A_y y e_3 + A_z e_4$ – 4-мерный электромагнитный потенциал;

$\Diamond = \frac{\partial}{\partial ct} e_1 + \frac{\partial}{\partial x} e_2 + \frac{\partial}{\partial y} e_3 + \frac{\partial}{\partial z} e_4$ – 4-мерный дифференциальный оператор.

Тогда, как показано в [24], система уравнений Максвелла сведется к такой системе из двух уравнений:

$$\Diamond \Psi = 0$$
$$\frac{d^2 \Phi}{dZ^2} = 2\pi J \cdot \qquad (2.4)$$

Система (2.4) намного проще, чем исходные системы уравнений (2.1) и (2.2).

Еще лучшие результаты дает применение кватернионов – некоммутативной гиперкомплексной системы четвертого порядка с таблицей умножения

H	e_1	e_2	e_3	e_4
e_1	e_1	e_2	e_3	e_4
e_2	e_2	$-e_1$	e_4	$-e_3$
e_3	e_3	$-e_4$	$-e_1$	e_2
e_4	e_4	e_3	$-e_2$	$-e_1$

(2.5)

В работе [25], эта связь выводится через матрицы Паули. Мы будем делать построения непосредственно с помощью кватернионного и евклидова базисов.

Кватернион можно представить в виде суммы скалярной и векторной частей:

$$a = a_1 e_1 + (a_2 e_2 + a_3 e_3 + a_4 e_4) = a_s + a_v \qquad (2.6)$$

Это представление возможно потому, что кватернионное умножение базисных элементов системы кватернионов (e_2, e_3, e_4) можно взаимно однозначно связать с векторным умножением ортов евклидова пространства $(\varepsilon_2, \varepsilon_3, \varepsilon_4)$.

Будем обозначать кватернионное произведение двух кватернионов значком $\otimes$. Тогда кватернионное произведение двух кватернионов a и b выглядит так:

$$a \otimes b = (a_s + a_v) \otimes (b_s + b_v) = a_s b_s - (a_v, b_v) + a_s b_v + a_v b_s + [a_v, b_v]. \qquad (2.7)$$

Необходимо указать, что скалярное (a_v, b_v) и векторное $[a_v, b_v]$ произведения векторных частей кватернионов имеют чисто формальный смысл, так как таких произведений для кватернионов нет. Кроме того, векторные части кватернионов заданы не в евклидовом базисе, а в кватернионном базисе.

Найдем соотношения между элементами векторной части кватернионного базиса (e_2, e_3, e_4) и элементами евклидового базиса $(\varepsilon_2, \varepsilon_3, \varepsilon_4)$:

$$ie_k \otimes ie_k = e_1 \Rightarrow 1 = [\varepsilon_k, \varepsilon_k],\ k = 2,3,4,$$

откуда можно получить связь между ортами кватернионного и евклидова пространств:

$$e_k \Rightarrow -i\varepsilon_k;\ \ \varepsilon_k \Rightarrow ie_k,\ k = 2,3,4. \qquad (2.8)$$

С учетом (2.8) кватернионное произведение двух кватернионов (2.7) в евклидовом базисе будет иметь вид:

$$a \otimes b \Rightarrow a_1 b_1 + (\tilde{a}, \tilde{b}) + a_1 \tilde{b} + \tilde{a} b_1 + i[\tilde{a}, \tilde{b}]. \qquad (2.9)$$

Здесь $\tilde{a}, \tilde{b}$ – векторные части кватернионов в евклидовом базисе, например: $\tilde{a} = -i(a_2\varepsilon_2 + a_3\varepsilon_3 + a_4\varepsilon_4)$.

Из соответствия (2.9) следует такой вывод, который понадобится далее: *если выражение имеет вид правой части (2.9), то оно соответствует кватернионному произведению некоторых кватернионов.*

Умножив второе уравнение системы (2.1) на мнимую единицу i, и сложив с первым с учетом того, что $divD = \nabla D$ и $divB = \nabla B$, получим:

$$\nabla(D + iB) = 4\pi\rho, \qquad (2.10)$$

где $D + iB = f$ вектор напряженности электромагнитного поля. То есть (2.10) принимает вид:

$$\nabla f = 4\pi\rho. \qquad (2.11)$$

Умножая третье уравнение системы (2.1) на мнимую единицу i, и складывая с четвертым, получим:

$$\frac{1}{c}\partial_1 f + [\nabla, (iE - H)] = -\frac{4\pi}{c} j. \qquad (2.12)$$

Так как в вакууме

$$iE - H = iD - B = i(D + iB) = if, \qquad (2.13)$$

то (2.12) примет вид

$$\frac{1}{c}\partial_1 f + i[\nabla, f] = -\frac{4\pi}{c} j. \qquad (2.14)$$

Суммируем уравнения (2.11) и (2.14):

$$\frac{1}{c}\partial_1 f + \nabla f + i[\nabla, f] = 4\pi\left(\rho - \frac{1}{c} j\right). \qquad (2.15)$$

Поставим в соответствие евклидовому вектору f кватернион Φ с нулевой скалярной частью, дифференциальному оператору $\left(\frac{1}{c}\partial_1 + \nabla\right)$ кватернион

производной $Д$, а выражению $\left(\rho - \frac{1}{c} j\right)$ – кватернион тока I. С учетом этого вся система уравнений Максвелла превратится в одно кватернионное уравнение

$$Д \otimes Ф = 4\pi I. \tag{2.16}$$

2.1.2 Моделирование поворота твердого тела в пространстве с помощью кватернионов

При умножении трехмерного вектора r слева на ортогональную матрицу [29] получается вектор r', модуль которого равен модулю исходного вектора r. Это обусловлено тем, что определитель ортогональной матрицы M равен 1. Таким образом, преобразование

$$r' = Mr, \tag{2.17}$$

или

$$\begin{vmatrix} r_1' \\ r_2' \\ r_3' \end{vmatrix} = \begin{vmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{vmatrix} \begin{vmatrix} r_1 \\ r_2 \\ r_3 \end{vmatrix} \tag{2.18}$$

представляет собой перемещение конца вектора r по сфере радиуса $|r|$ в другую точку той же сферы, или, вращение вектора r вокруг некоторой оси с единичным ортом u на угол θ. Но из матрицы M не видно, какой оси u и углу поворота θ она соответствует. С другой стороны, на практике очень часто задача формулируется именно так: найти матрицу M такую, чтобы она поворачивала любой вектор r около оси $u(u_1, u_2, u_3)$ на угол θ. Как показано в работах [28, 30], модель поворота будет иметь довольно громоздкий вид:

$$\begin{vmatrix} r_1' \\ r_2' \\ r_3' \end{vmatrix} = \begin{vmatrix} cos\theta + (1-cos\theta)u_1^2 & (1-cos\theta)u_1u_2 - (sin\theta)u_3 & (1-cos\theta)u_1u_3 + (sin\theta)u_2 \\ (1-cos\theta)u_1u_2 + (sin\theta)u_3 & cos\theta + (1-cos\theta)u_2^2 & (1-cos\theta)u_2u_3 - (sin\theta)u_1 \\ (1-cos\theta)u_1u_3 - (sin\theta)u_2 & (1-cos\theta)u_2u_3 + (sin\theta)u_1 & cos\theta + (1-cos\theta)u_3^2 \end{vmatrix} \begin{vmatrix} r_1 \\ r_2 \\ r_3 \end{vmatrix}. \tag{2.19}$$

Как показано в [26–28, 30], если ввести кватернион вращения

$$q = cos\frac{\theta}{2} + u\, sin\frac{\theta}{2}, \tag{2.20}$$

то оператор вращения вектора r вокруг оси с единичным ортом u на угол θ имеет очень простой вид:

$$r' = qrq^{-1}, \tag{2.21}$$

где $q^{-1} = cos\frac{\theta}{2} - u\, sin\frac{\theta}{2}$ – сопряженный кватернион.

Вращение происходит по часовой стрелке, если рассматривать вращение по направлению вектора u.

Модель (2.21) замечательна еще и тем, что она позволяет «складывать» повороты: если нужно повернуть сначала на кватернион q, а потом – на другой кватернион p, то достаточно выполнить:

$$r' = pqrq^{-1}p^{-1}. \tag{2.22}$$

Здесь сначала выполняется q, а потом p. Порядок имеет значение. И таких поворотов можно выполнять сколько угодно. Таким образом, легко представить, насколько упрощается модель поворота по сравнению с применением ортогональной матрицы.

2.1.3 Передаточная функция цифрового реверсивного фильтра

Применение гиперкомплексных числовых систем для синтеза структур цифровых фильтров подробно рассматривается в работах [1, 31–33]; там, в частности, показано, что цифровые фильтры с гиперкомплексными параметрами имеют большее быстродействие и лучшие характеристики по интегральной параметрической чувствительности.

Передаточная функция цифрового фильтра n-го порядка с вещественными параметрами имеет вид дробно-рациональной функции от переменной $z = e^{j\omega}$:

$$H_R = \frac{\sum_{k=0}^{n} \varphi_k z^{-k}}{\sum_{k=1}^{n} \phi_k z^{-k} + 1}, \tag{2.23}$$

где: $\phi_k, \varphi_k \in R$; ω – угловая частота.

Преобразование передаточной функции цифрового фильтра с действительными параметрами в передаточную функцию с гиперкомплексными параметрами базируется на следующих соображениях.

Рассмотрим гиперкомплексную дробно-рациональную функцию $R(z) = \frac{P(z)}{Q(z)}$ от переменной z.

$$R(z) = \frac{P(z)}{Q(z)}. \tag{2.24}$$

Здесь:

$$P(z) = \sum_{i=0}^{l} p_i z^{-i},\ Q(z) = \sum_{i=0}^{l} q_i z^{-i},\ p_i, q_i \in \Gamma(e,m),\ q_0 = 1, \tag{2.25}$$

где $\Gamma(e,m)$ есть гиперкомплексная числовая система размерности m.

У функции (2.25) и в числителе и в знаменателе присутствуют базисные элементы базиса $e = \{e_1, ..., e_m\}$ гиперкомплексной числовой системы $\Gamma(e,m)$.

Приведем функцию (2.25) к виду гиперкомплексной функции. Напомним, что гиперкомплексная функция имеет форму [1]:

$$f(X)=\sum_{i=1}^{m} f_i(x_1,...,x_m)\cdot e_i, \qquad X=\sum_{i=1}^{m} x_i e_i \in \Gamma,\ x_i \in R.$$

Теоретической основой такого приведения является изложенный в [1] алгоритм деления двух гиперкомплексных чисел. Для его реализации необходимо числитель и знаменатель функции (2.24) умножить на полином, сопряженный к знаменателю. Пусть сопряженный полином имеет вид:

$$\overline{Q(z)}=\sum_{i=0}^{l} \tilde{q}_i z^{-i}. \tag{2.26}$$

Произведение $Q(z)\cdot\overline{Q(z)}$ есть норма полинома $Q(z)$, умноженная на единичный элемент ε системы $\Gamma(e,m)$:

$$Q(z)\cdot\overline{Q(z)}=N(Q(z))\cdot\varepsilon. \tag{2.27}$$

Здесь $N(Q(z))\in R$.

Норма гиперкомплексного числа есть форма m-ой степени относительно компонентов этого числа. Так как $Q(z)$ – полином l-ой степени, то норма $N(Q(z))$ будет формой $l\cdot m$-ой степени относительно оператора z.

Соответственно и сопряженное число $\overline{Q(z)}$ будет формой степени $l(m-1)$, а произведение $P(z)\cdot\overline{Q(z)}$ будет формой степени m относительно элементов гиперкомплексного числа, а относительно оператора z – степени $m\cdot l$.

Таким образом, функция (2.24) превращается в

$$R(z)=\frac{P(z)\cdot\overline{Q(z)}}{N(Q(z))}, \tag{2.28}$$

которую можно представить в виде гиперкомплексной функции:

$$\frac{P(z)\cdot\overline{Q(z)}}{N(Q(z))}=\sum_{i=1}^{m}\frac{f_i(z)}{N(Q(z))}\cdot e_i=\sum_{i=1}^{m} R_i(z)\cdot e_i. \tag{2.29}$$

Полиномы, стоящие в числителях и знаменателях компонент гиперкомплексной функции (2.29), параметрически зависят от коэффициентов полиномов (2.25).

Так как в соответствии с (2.25) $q_0=1$, то можно представить знаменатель (2.29) в виде:

$$N(Q(z))=N^{*}(Q(z))+\varepsilon. \tag{2.30}$$

Представим, что (2.23) – передаточная функция цифрового фильтра, в котором информация имеет гиперкомплексную структуру и преобразуется по законам выполнения гиперкомплексных операций. Передаточная функция такого фильтра будет также гиперкомплексная:

$$R(z)=R_1(z)\cdot e_1+R_2(z)\cdot e_2+...+R_m(z)\cdot e_m, \tag{2.31}$$

образуя по каждой размерности свой действительный фильтр.

Пусть задан цифровой фильтр $l\cdot m$-ого порядка с действительной передаточной функцией

$$H = \frac{\sum_{i=0}^{ml} \varphi_i z^{-i}}{\sum_{i=1}^{ml} \phi_i z^{-i} + 1} \tag{2.32}$$

Выбирая подходящим образом для $k \in \overline{1,...,m \cdot l}$ такой набор параметров полиномов $f_k(z)$ и $N^*(Q(z))$, что

$$R_k(z) = \frac{f_k(z)}{N^*(Q(z)) + \varepsilon} = \frac{\sum_{i=0}^{ml} \varphi_i z^{-i}}{\sum_{i=1}^{ml} \phi_i z^{-i} + 1}, \tag{2.33}$$

и, полагая для определенности без ограничения общности $k = 1$, мы получим, что фильтр с передаточной функцией (2.31) фильтрует входной сигнал по каналу первой размерности точно так же, как и фильтр с передаточной функцией (2.33). На выходе полезным будет также сигнал по каналу первой размерности. Но первый фильтр имеет порядок, равный l, а второй – ml.

Таким образом, действительную передаточную функцию вида (2.23) можно представить более простой гиперкомплексной передаточной функцией вида

$$H_{\Gamma} = \frac{A + Bz^{-1}}{\varepsilon + Cz^{-1}}, \tag{2.34}$$

где $A, B, C, \in \Gamma(e,m)$.

2.2 Уменьшение объема вычислений с помощью методов ГЧС

Использование методов гиперкомплексных числовых систем в математическом моделировании может привести не только к упрощению вида модели, но и сокращению количества вычислений при ее создании. В данном подразделе будут рассмотрены некоторые практические задачи, в которых использование ГЧС значительно снижает количество вычислений.

2.2.1 Применение кватернионов для описания движения твердого тела в пространстве

Рассмотрим сформулированную выше задачу о повороте точки вокруг оси. Пусть в трехмерном пространстве задана точка $x(x_1, x_2, x_3)$. Требуется найти ее координаты $y(y_1, y_2, y_3)$ после поворота вокруг оси U на угол θ. При этом ось U задается единичным ортом u с компонентами (u_1, u_2, u_3). Для решения этой задачи необходимо вектор x умножить слева на матрицу поворота M:

$$y = Mx. \tag{2.35}$$

Компоненты матрицы поворота M являются функциями параметров поворота, как это видно из выражения (2.19). Для вычисления всех компонентов

этой матрицы необходимо выполнить 16 умножений, 11 сложений и определить 1 значение тригонометрической функции.

Эту же задачу можно решать с помощью кватернионов по формуле (2.21), куда входит кватернион $q = cos\frac{\theta}{2} + u\,sin\frac{\theta}{2}$, для вычисления компонент которого необходимо сделать всего лишь 4 умножения, 1 сложение и, как и ранее, определить 1 значение тригонометрической функции. То есть этот метод гораздо проще первого.

Правда, следует отметить, что при вычислениях новых координат повернутой точки ситуация меняется на противоположную: по формуле (2.35) необходимо выполнить 9 умножений, а с помощью кватернионов, используя методы ускоренного умножения кватернионов, изложенные в работах [34–36], – 14 умножений. Таким образом, можно сделать вывод, что при выполнении обоих этапов решения задачи – и построения матрицы поворота или кватерниона поворота и проведения самого поворота – использование кватернионов снижает количество вычислений.

Рассмотрим более сложную задачу – задачу «сложения» поворотов. Пусть ту же точку $x(x_1, x_2, x_3)$ требуется повернуть сначала вокруг оси U_1 на угол θ_1, а потом вокруг другой оси U_2 на угол θ_2.

По первому методу требуется найти две матрицы поворота, соответствующие двум осям, и перемножить их, в результате найти общую матрицу поворота. Учитывая, что для умножения двух матриц размерами 3×3 требуется $3^3 = 27$ умножений, всего потребуется 59 умножений.

Для вычисления кватернионов поворотов, входящих в (2.22), необходимо выполнить всего 8 умножений, а всего вычислений в (2.22) – 24 умножения, что гораздо эффективнее первого метода.

2.2.2 Использование изоморфных переходов

Другим перспективным направлением снижения количества вычислений при оперировании с гиперкомплексными числами является использование ГЧС со слабо-заполненными таблицами умножения. В качестве примера такой ГЧС можно привести наиболее простую в этом смысле ГЧС – систему двойных чисел в виде прямой суммы $W_1(f,2) = R \oplus R$:

W_1	f_1	f_2
f_1	f_1	0
f_2	0	f_2

Как видим, в этой таблице половина ячеек – это нули, Поэтому она называется слабо-заполненной.

Однако при математическом моделировании непосредственное использование слабо-заполненных ГЧС как правило неэффективно, так как уравнения моделей содержат слишком мало параметров. Поэтому целесообразно использовать ГЧС, изоморфные слабо-заполненным системам, но с сильным

заполнением. Так, для приведенной ГЧС $W_1(f,2)=R\oplus R$ изоморфной, но сильно-заполненной ГЧС будет система двойных чисел $W(e,2)$ с таблицей умножения:

W	e_1	e_2
e_1	e_1	e_2
e_2	e_2	e_1

Оператор изоморфизма имеет вид [1]:

$$L:\begin{cases} e_1 = f_1 + f_2 \\ e_2 = f_1 - f_2 \end{cases},\ L^{-1}:\begin{cases} f_1 = (e_1 + e_2)/2 \\ f_2 = (e_1 - e_2)/2 \end{cases}. \tag{2.36}$$

Как видно из этих таблиц, при выполнении операций сложения и умножения в системе $W(e,2)$ в два раза больше, чем в системе $W_1(f,2)=R\oplus R$. Поэтому целесообразно модель строить в системе $W(e,2)$, а вычисления проводить в системе $W_1(f,2)=R\oplus R$ с последующим обратным переходом в систему $W(e,2)$. Для перехода используется оператор перехода (2.36).

При увеличении размерности задачи этот эффект будет только увеличиваться. В следующем разделе будет сделана оценка его для задач разного типа, а далее рассмотрим его проявление на примере одной конкретной задачи.

Рассмотрим вычисления, которые нужно производить при функционировании цифрового фильтра третьего порядка, реализованного с помощью фильтра первого порядка с передаточной функцией

$$H_T = \frac{A + Bz^{-1}}{e_1 + Cz^{-1}}, \tag{2.37}$$

коэффициенты которой принадлежат системе триплексных чисел

$$A, B, C \in T,$$

таблица умножения которой имеет следующий вид:

T	e_1	e_2	e_3
e_1	e_1	e_2	e_3
e_2	e_2	$(e_3 - e_1)/2$	$-e_2$
e_3	e_3	$-e_2$	e_1

Структура фильтра показана на рис.2.1. Фильтр описывается дифференциально-разностным уравнением

$$y[n] = Ax[n] + Bx[n-1] - Cy[n-1]. \tag{2.38}$$

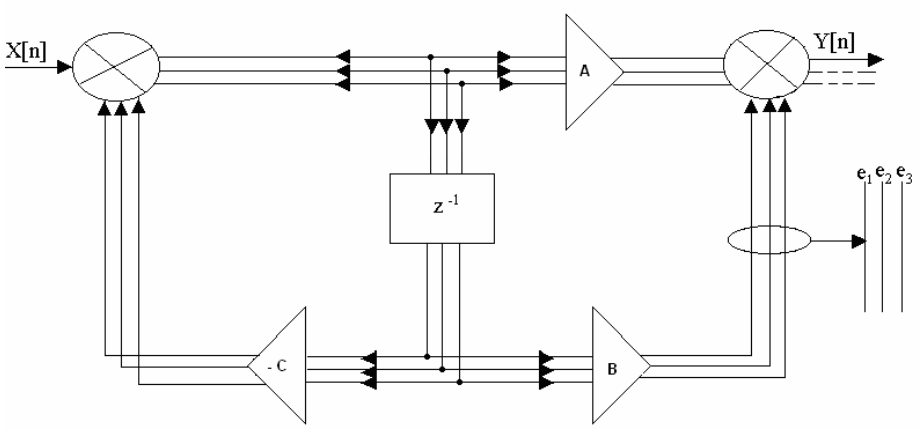


Рис.2.1 Структурная схема фильтра с триплексными параметрами

Работа триплексного фильтра сводится к рекуррентным вычислениям по разностному уравнению (2.38). Для этого необходимо выполнить 3 гиперкомплексных умножения, каждое из которых в системе триплексных чисел T сводится к 10 вещественным умножениям. Для уменьшения количества вычислений целесообразно делать их не в системе триплексных чисел $T(e,3)$, а в изоморфной ей системе действительно-комплексных чисел $R \oplus C(f,3)$, таблица умножения которой имеет вид:

$R \oplus C$	f_1	f_2	f_3
f_1	f_1	0	0
f_2	0	f_2	f_3
f_3	0	f_3	$-f_2$

(2.39)

Оператор изоморфизма между системами $T(e,3)$ и $R \oplus C(f,3)$ имеет вид:

$$L: \begin{cases} e_1 = f_1 + f_2 \\ e_2 = \pm f_3 \\ e_3 = f_1 - f_2 \end{cases}, \qquad L^{-1}: \begin{cases} f_1 = (e_1 + e_3)/2 \\ f_2 = (e_1 - e_3)/2 \\ f_3 = \pm e_2. \end{cases} \qquad (2.40)$$

Как видно из таблицы умножения (2.39), умножение чисел в системе $R \oplus C(f,3)$ требует 5 вещественных умножений. Кроме того, существует алгоритм умножения комплексных чисел, требующий 3-х вещественных умножений [37]:

$$(a+bi)(c+di) = (a-b)d + a(c-d) + [(a-b)d + b(c+d)]i. \qquad (2.41)$$

Поэтому умножение в системе $R \oplus C(f,3)$ требует всего лишь 4 вещественных умножения. Это оправдывает целесообразность выполнения гиперкомплексного умножения в уравнении (2.38) путем перехода к изоморфной ГЧС.

ГЛАВА 3 СЛОЖНОСТЬ ВЫЧИСЛЕНИЙ С ИСПОЛЬЗОВАНИЕМ ИЗОМОРФНЫХ ГЧС

В данной главе рассматривается изменение количества вычислений при переходе от действий в сильно-заполненных ГЧС к изоморфным им слабо-заполненным. При этом будет оцениваться количество операций над вещественными числами, необходимых для выполнения какой-либо процедуры над гиперкомплексными числами. В дальнейшем для краткости будем применять термины «вещественная операция» и «гиперкомплексная операция». Этот эффект во многом зависит от конкретно рассматриваемых ГЧС. Кроме того, нужно учитывать и те дополнительные операции, которые необходимы для перехода от одной ГЧС к другой и обратно. При выполнении гиперкомплексной операции сложения количество вещественных сложений не зависит от заполненности таблиц умножения ГЧС и равно размерности рассматриваемых систем. Поэтому в дальнейшем гиперкомплексное сложение рассматривать не будем.

3.1 Гиперкомплексное умножение

Рассмотрим идеальный случай, когда одна из двух изоморфных ГЧС имеет полностью заполненную каноническую таблицу умножения, а вторая – каноническую таблицу диагонального вида, в которой заполнены только ячейки, находящиеся на главной диагонали.

Тогда гиперкомплексное умножение в первом случае сводится к n^2 вещественным умножениям и $n(n-1)$ вещественным сложениям. Во втором случае, соответственно, n вещественных умножений и 0 вещественных сложений. То есть эффект очень велик – только вещественных умножений в n раз меньше. Количество вещественных операций, необходимых для перехода от одной ГЧС к другой и обратно зависит от конкретного типа этих ГЧС. Поэтому целесообразно рассмотреть конкретные типы ГЧС. Мы будем рассматривать не отдельные типы ГЧС, а ряды пар изоморфных ГЧС, генерируемых путем применения процедуры умножения размерности [2].

3.1.1 Ряды изоморфных ГЧС, полученных последовательным удвоением ГЧС W и W_1

Рассмотрим две последовательности попарно изоморфных ГЧС типа

$$Д\left(W^{(1)},W^{(2)},...,W^{(n)},2^n\right)\simeq Д\left(W_1^{(1)},W_1^{(2)},...,W_1^{(n)},2^n\right),\ n=1,2,...,$$

где для упрощения выражений в операторе умножения размерности $Д$ приведен ряд ГЧС, для которых применяется процедура умножения размерности последовательно, а в конце приводится размерность полученной ГЧС. Порядок ГЧС в списке не имеет значения, так как в соответствии с Теоремой 1 [2] операнды коммутируют.

ГЧС $Д\left(W^{(1)},W^{(2)},...,W^{(n)},2^n\right)$ имеет полностью заполненную таблицу умножения, а в таблице умножения ГЧС $Д\left(W_1^{(1)},W_1^{(2)},...,W_1^{(n)},2^n\right)$ заполнена только главная диагональ.

Рассчитаем количество вещественных операций при гиперкомплексном умножении в этих ГЧС.

Введём следующие обозначения:

n – порядок процедуры умножения размерности. Если $n=0$, то процедура умножения размерности не применялась; $N=2^{n+1}$ размерность полученной ГЧС.

Тогда число вещественных умножений при гиперкомплексном умножении в системе $Д\left(W_1^{(1)},W_1^{(2)},...,W_1^{(n)},2^n\right)$ равно

$$M_1=N^2=2^{2(n+1)}.$$

Число вещественных сложений

$$S_1=N(N-1)=2^{n+1}\left(2^{n+1}-1\right).$$

Соответственно для ГЧС $Д\left(W_1^{(1)},W_1^{(2)},...,W_1^{(n)},2^n\right)$:

$$M_2=N=2^{n+1}.$$

$$S_2=0.$$

Определим число вещественных сложений $S_{пер}$ при переводе гиперкомплексных чисел из одной ГЧС в другую и обратно. В соответствии с операторами изоморфизма таких ГЧС [2]

$$S_{пер}=2S_1.$$

При этом деление в переходе на 2^{n+1} можно не учитывать, так как это короткая операция. Таким образом, в данном случае число вещественных умножений уменьшается в $\left(2^{n+1}-1\right)$ раз, но число сложений увеличивается в ≈ 2 раза.

3.1.2 Ряды изоморфных ГЧС на основе системы комплексных чисел C

Рассмотрим изученные в разделе 1.3 следующие ряды изоморфных пар ГЧС:

1) $Д\left(C,W^{(1)},W^{(2)},...,W^{(n)},2^{n+1}\right)\simeq Д\left(C,W_1^{(1)},W_1^{(2)},...,W_1^{(n)},2^{n+1}\right)=\overset{2^n}{\underset{i=1}{\oplus}}C^{(i)};$

2) $Д\left(C,C^{(1)},C^{(2)},...,C^{(n)},2^{n+1}\right)=Д\left(K,C^{(2)},...,C^{(n)},2^{n+1}\right)\simeq\overset{2^n}{\underset{i=1}{\oplus}}C^{(i)};$

3) $Д\left(K,W^{(1)},W^{(2)},...,W^{(n)},2^{n+2}\right)\simeq Д\left(K,W_1^{(1)},W_1^{(2)},...,W_1^{(n)},2^{n+2}\right)=\overset{2^{n+1}}{\underset{i=1}{\oplus}}C^{(i)};$

4) $Д\left(K,K^{(1)},K^{(2)},...,K^{(n)},2^{2(n+1)}\right)\simeq Д\left(C\oplus C,(C\oplus C)^{(1)},...,(C\oplus C)^{(n)},2^{2(n+1)}\right)=\overset{2^{2n+1}}{\underset{i=1}{\oplus}}C^{(i)}.$

Здесь левые от значка изоморфизма операторы умножения размерности дают различные ГЧС с полностью заполненными таблицами умножения. Их размерности обозначим через N. Размерность N есть некоторая степень двой-

ки. Эти ГЧС изоморфны прямой сумме из $\frac{N}{2}$ слагаемых, представляющих собой системы комплексных чисел $C^{(i)}$.

Структура таблиц умножения таких ГЧС показана на рис.3.1.

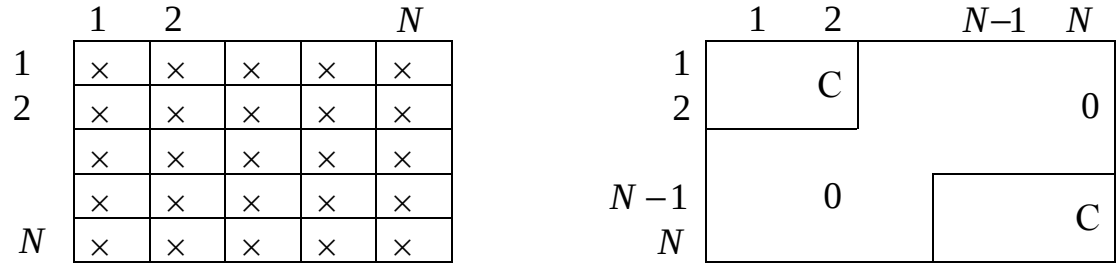


Рис.3.1 Структура таблиц умножения ГЧС. (слева –полностью заполненная, справа – диагонального вида)

По этим рисункам легко рассчитать количество вещественных операций, необходимых для перемножения гиперкомплексных чисел: умножений (M_1 и M_2 соответственно) и сложений (S_1 и S_2 соответственно):

$$M_1 = N^2, \quad M_2 = 4 \cdot \frac{N}{2} = 2N .$$

$$S_1 = N(N-1), \quad S_2 = 2 \cdot \frac{N}{2} = N .$$

Здесь необходимо учесть и те дополнительные операции сложения, которые необходимы для перевода чисел из одной ГЧС в другую и обратно. Как следует из раздела 1.3, их количество будет

$$S_П = 2\left(\frac{N}{2} - 1\right)N .$$

Общее количество сложений:

$$S_\Sigma = S_2 + S_П = N(N-1).$$

Значит, при переходе от ГЧС с сильно-заполненной таблицей умножения к ГЧС диагонального вида при умножении гиперкомплексных чисел количество вещественных умножений уменьшится в $\frac{M_1}{M_2} = \frac{N}{2}$ раз, а количество вещественных сложений не изменяется даже при учете перехода из одной ГЧС в другую и обратно.

3.2 Расчет норм, сопряженных чисел и нелинейностей

Рассмотрим на конкретных примерах, насколько изменяется количество вещественных операций при вычислении норм, сопряжений и нелинейностей в результате перехода от ГЧС с сильно-заполненными таблицами умножения к

ГЧС со слабо-заполненными таблицами. Пусть, как и ранее, гиперкомплексное число A имеет вид $A=\sum_{i=1}^{n} a_i e_i$.

3.2.1 Вычисление норм

1) Системы W и W_1 размерности $n=2$.

Нормы в этих системах определяются по формулам (здесь и далее формулы для норм см. Приложение 10.2 из [2]);

$$N_W(A)=a_1^2-a_2^2;\ \ N_{W_1}(A)=a_1\cdot a_2.$$

Как видно из этих формул, в первом случае нужно выполнить 2 вещественных умножения и 1 сложение, а во втором случае – 1 умножение и 0 сложений. Необходимо также учесть, что для перехода от одной системы к другой и обратно потребуется 4 сложения (2.16).

Таким образом, в ГЧС W с сильно-заполненной таблицей умножения при вычислении нормы необходимо выполнить 2 умножения и 1 сложение, а в ГЧС W_1 со слабо-заполненной таблицей умножения – соответственно 1 и 4, то есть здесь эффект сравнительно невелик.

2) Системы T и $R\oplus C$ размерности $n=3$.

$$N_T(A)=(a_1+a_3)\left(a_1^2+a_2^2+a_3^2-2a_1a_3\right);\qquad N_{R\oplus C}(A)=a_1\left(a_2^2+a_3^2\right).$$

Число умножений в обеих системах равно соответственно 5 и 3, сложений – 4 и 1. Для перехода от одной системы к другой и обратно потребуется 4 сложения (2.18). Итого:

$$M_T=5;\qquad M_{R\oplus C}=3;\ \ S_T=3;\qquad S_{R\oplus C}=4.$$

Здесь эффективность применения процедуры перехода явная.

3) Системы K и $C\oplus C$ размерности $n=4$.

$$N_K(A)=\left(\sum_{i=1}^{4}a_i^2+2(a_2a_3-a_1a_4)\right)\left(\sum_{i=1}^{4}a_i^2-2(a_2a_3-a_1a_4)\right);$$

$$N_{C\oplus C}(A)=\left(a_1^2+a_2^2\right)\left(a_3^2+a_4^2\right).$$

Учитывая, что переход от одной системы к другой требует 8 сложений, получим:

$$M_K=7;\qquad M_{R\oplus C}=5;\ \ S_K=6;\qquad S_{R\oplus C}=10.$$

3.2.2 Вычисление сопряженных чисел

Ограничимся рассмотрением систем триплексных чисел K и бикомплексных чисел – прямой суммы $C\oplus C$. В соответствии с приложением 10.3 из [2] сопряженные числа в этих системах имеют вид:

$$\overline{A}_{\text{K}}=\left((a_1+a_4)^2+(a_2-a_3)^2\right)\left((a_1-a_4)e_1+(a_2+a_3)e_2\right)+\left((a_1-a_4)^2+(a_2+a_3)^2\right)a_2-a_3\left((a_1+a_4)e_3-(a_2-a_3)e_4\right);$$

$$\overline{A}_{C\oplus C}=\left(a_1e_1-a_2e_2\right)\left(a_3^2+a_4^2\right)+\left(a_3e_3-a_4e_4\right)\left(a_1^2+a_2^2\right).$$

Переход из одной системы в другую и обратно, как это следует из подраздела 1.2.4, требует 8 сложений. Таким образом:

$$M_K = 8; \quad M_{R\oplus C} = 8; \quad S_K = 10; \quad S_{R\oplus C} = 10.$$

То есть, здесь никакого эффекта нет.

3.2.3 Вычисление экспоненциальных функций

Как показывает анализ представлений экспонент (Приложение 10.4 [2]), основной объем этих вычислений в гиперкомплексной области занимает вычисление экспоненциальных и тригонометрических функций от вещественного аргумента. Поэтому будем обращать внимание только на вычисление этих трансцендентных функций.

1) Системы W и W_1 размерности $n = 2$.

Представления экспонент в этих системах определяются соответственно по формулам (здесь и далее Приложение 10.4 [2]):

$$Exp_W(A) = e^{a_1}(\cos a_2 \cdot e_1 + \sin a_2 \cdot e_2); \quad Exp_{W_1}(A) = e^{a_1}e_1 + e^{a_2}e_2.$$

Для системы W_1 нужно вычислять на одну трансцендентную функцию меньше.

2) Системы T и $R \oplus C$ размерности $n = 3$

$$Exp_T(A) = \frac{1}{2}\left(e^{a_1+a_3} + e^{a_1-a_3}\cos a_2\right)e_1 + e^{a_1-a_3}\sin a_2 \cdot e_2 + \frac{1}{2}\left(e^{a_1+a_3} - e^{a_1-a_3}\cos a_2\right)e_3$$

$$Exp_{R\oplus C}(A) = e^{a_1}e_1 + e^{a_2}(\cos a_3 \cdot e_2 + \sin a_3 \cdot e_3).$$

Здесь количество вычислений трансцендентных функций одинаково и равно четырем.

3) Системы K и $C \oplus C$ размерности $n = 4$.

Как видно из Приложения 10.4 [2], количество вычислений трансцендентных функций в случае применения ГЧС $C \oplus C$ на одно меньше (в системе K – 7, в системе $C \oplus C$ – 6).

Из всего вышеприведенного можно сделать следующие выводы:

1. При выполнении операции умножения гиперкомплексных чисел переход от ГЧС с сильно-заполненными таблицами умножения к ГЧС со слабо-заполненными таблицами дает ощутимый выигрыш в количестве вычислений над вещественными числами.
2. В случае выполнения остальных операций этот эффект зависит от конкретно применяемых ГЧС.

ГЛАВА 4 БЫСТРЫЕ АЛГОРИТМЫ ВЫЧИСЛЕНИЯ СВЕРТКИ

4.1 Основные определения

Цифровой фильтр с импульсной характеристикой конечной длины (КИХ-фильтр) является одним из наиболее важных устройств для решения задач цифровой обработки информации. Выходная последовательность КИХ-фильтра является линейной сверткой входной последовательности с последовательностью значений импульсной характеристики КИХ-фильтра. Так как эта операция производится много раз, то весьма актуальной является задача синтеза быстрых алгоритмов выполнения линейной свертки, чему и посвящена настоящая глава.

4.1.1 Свертка и корреляция

В математическом анализе существует операция сворачивания двух функций $x(t)$ и $y(t)$. По определению сверткой этих функций будет такая функция $S(\tau)$, что

$$S(\tau)=\int_{-\infty}^{+\infty} f(t-\tau)\,y(t)\,dt\,. \tag{4.1}$$

Пределы интегрирования могут быть конечные. Заменой переменной интегрирования можно показать, что подынтегральные функции в (4.1) коммутируют. Символически свертку записывают так:

$$S = x * y = y * x\,. \tag{4.2}$$

Дискретный аналог выражения (4.1) имеет вид:

$$S[n]=\sum_{k=-\infty}^{+\infty} x[n-k]\cdot y[k]. \tag{4.3}$$

Здесь последовательность $y[k]$ будем называть ядром свертки. В некоторых задачах приходится определять не свертку последовательностей, а корреляцию, определяемую формулой:

$$K[n]=\sum_{k=-\infty}^{+\infty} x[n+k]\cdot y[k]. \tag{4.4}$$

Формулы для свертки и корреляции тесно связаны. Действительно, если в (4.4) сделать замену переменной суммирования k на $-k$ и положить $\tilde{y}[k]=y[-k]$, то есть развернуть относительно нулевой точки, то получится формула свертки (4.3). Поэтому в дальнейшем мы будем синтезировать быстрые алгоритмы либо для свертки, либо для корреляции, зная, что они могут быть легко преобразованы друг в друга.

Обычно последовательности сигнала и ядра конечны. Пусть сигнал

$$x[n]\begin{cases} \neq 0 : n\in[0,N-1] \\ = 0 : n\in(-\infty,-1]\cup[N,+\infty) \end{cases},$$

то есть имеет всего N ненулевых значений, а ядро

$$y[y]\begin{cases} \neq 0 : n \in [-m_1, m_2] \\ = 0 : n \in (-\infty, -m_1 - 1] \cup [m_2 + 1, +\infty) \end{cases},$$

то есть длина ядра равна $m_1 + m_2 + 1 = M$ отсчетов. Если эти длины последовательностей сигнала и ядра подставить в формулу свертки (4.3), то получим:

$$S[n]\begin{cases} \neq 0 : n \in [-m_1, N - 1 + m_2] \\ = 0 : n \in (-\infty, -m_1 - 1] \cup [N + m_2, +\infty) \end{cases},$$

то есть свертка имеет $N + M - 1$ ненулевых отсчетов. Можно сказать, что входная последовательность $x[n]$ сверткой с $y[n]$ продлевается на $M - 1$ отсчет.

4.1.2 Линейная и циклическая свертки

Свертка вида (4.3) называется линейной сверткой. Кроме линейной свертки мы еще рассмотрим построение быстрых алгоритмов циклической свертки, которая определяется формулой [38–39]:

$$C[i] = \sum_{k=0}^{n-1} x[(i-k)(mod\ n)] \cdot y[k],\ i = 0, ..., n-1. \qquad (4.5)$$

Следуя [38], обозначим громоздкий индекс в (4.5) двойными скобками:

$$(i-k)(mod\ \ n) = ((i-k)). \qquad (4.6)$$

Очевидно:

$$0 \le ((i-k)) \le n. \qquad (4.7)$$

Ввиду (4.7) в (4.5) каждый отсчет ядра умножается на ненулевое значение входной последовательности, а каждый отсчет $C[i]$ состоит из n слагаемых. Таким образом, циклическая свертка определяется по формуле

$$C[i] = \sum_{k=0}^{n-1} x[((i-k))] \cdot y[k],\ i = 0, ..., n-1. \qquad (4.8)$$

Между линейной и циклической свертками существует тесная связь [38]. Используя её, можно вычислять линейные свертки длинных последовательностей с помощью быстрых алгоритмов циклической свертки коротких последовательностей.

Как отмечалось ранее в подразделе 1.1.5, числовую последовательность можно изобразить полиномом. Если последовательности сигналов будут:

$$\bar{a} = \{a[0], a[1], ..., a[n-1]\} \ \text{ и } \ b = \{b[0], b[1], ..., b[n-1]\}. \qquad (4.9)$$

Им можно сопоставить полиномы $A(x) = \sum_{i=0}^{n-1} a[i]x^i$; $B(x) = \sum_{i=0}^{l-1} b[i]x^i$, степени которых $(n-1)$ и $(l-1)$ соответственно. Тогда их произведение будет полиномом $(n+l-2)$-й степени:

$$S(x) = B(x)A(x) = \sum_{k=0}^{n+l-2} a[k] \cdot b[i-k]x^k. \qquad (4.10)$$

Коэффициенты полинома (4.10) и будут отсчетами линейной свертки исходных последовательностей сигналов $\bar{a}$ и $\bar{b}$, в чем легко убедиться, перемно-

жая полиномы по правилу их умножения. Из выражения (4.10) видно, что последовательности $\bar{a}$ и $\bar{b}$ здесь играют симметричную роль, то есть в выражении свертки их можно коммутировать.

Циклическая свертка двух последовательностей сигналов длиной n каждая в полиномиальном представлении определяется по формуле [38–39]:

$$C(x) = B(x)A(x) \ (mod\ x^n - 1).$$

4.2 Основные методы быстрого вычисления свертки

Существует большое количество методов быстрого вычисления линейной и циклической свертки. С некоторыми из них можно познакомиться в работах [37–39, 46] и многих других. Мы не ставили перед собой задачу дать подробный обзор этих методов. Здесь будут рассмотрены только основные методы. Особое внимание уделяется анализу тех моментов, где используется переход в такие пространства, в которых необходимые вычисления значительно упрощаются.

4.2.1 Алгоритм Кука-Тоома

Алгоритм вычисления линейной свертки Кука-Тоома [37–39] основан на представлении сворачиваемых последовательностей полиномами и восстановлении результирующего полинома-произведения с помощью интерполяционной формулы Лагранжа. При этом предполагается, что одна из сворачиваемых последовательностей является постоянной. Это условие является существенным, так как в противном случае алгоритм Кука-Тоома теряет свои преимущества в количестве операций.

Рассмотрим линейную свертку двух последовательностей $\bar{a} = \{a[0], ..., a[n-1]\}$ и $\bar{b} = \{b[0], ..., b[l-1]\}$:

$$\bar{c} = \bar{a} * \bar{b} = \{c[0], c[1], ..., c[n+l-2]\}, \tag{4.11}$$

где:

$$c[i] = \sum_{k=0}^{n-1} a[k]b[i-k]. \tag{4.12}$$

Рассмотрим также два полинома $A(x)$ и $B(x)$, степени которых $(n-1)$ и $(l-1)$ соответственно:

$$A(x) = \sum_{i=0}^{n-1} a[i]x^i; \ B(x) = \sum_{i=0}^{l-1} b[i]x^i. \tag{4.13}$$

Тогда их произведение будет полиномом $(n+l-2)$-й степени:

$$C(x) = B(x)A(x) = \sum_{k=0}^{n+l-2} c[k]x^k = \sum_{k=0}^{n+l-2} a[k]b[i-k]x^k, \tag{4.14}$$

коэффициенты которого $c[i]$ и будут отсчетами линейной свертки (4.11). Если возможно восстановление его каким-либо другим методом, то задача вычисления линейной свертки будет решена.

Так как $C(x)$ – полином $(n+l-2)$-й степени, то для точного восстановления полинома необходимо иметь значения его в $(n+l-1)$-й точке. Пусть этими точками будут $\gamma_0, \gamma_1, ..., \gamma_{n+l-2}$. По ним можно вычислить $(n+l-1)$ значения полиномов (4.13), а, значит и соответствующее количество значений полинома-свертки $C(x)$:

$$C(\gamma_i) = A(\gamma_i) \cdot B(\gamma_i), \qquad i = 0,1,...,n+l-2. \tag{4.15}$$

Восстановление полинома $C(x)$ можно проводить по интерполяционной формуле Лагранжа:

$$C(x) = \sum_{i=0}^{n+l-2} C(\gamma_i) \cdot L_i(x), \tag{4.16}$$

где $L_i(x)$ – интерполяционные полиномы Лагранжа, вычисляемые в данном случае по формуле:

$$L_i(x) = \frac{\prod_{j=0\ j\neq i}^{l+n-2} (x-\gamma_j)}{\prod_{j=0\ j\neq i}^{l+n-2} (\gamma_i-\gamma_j)}. \tag{4.17}$$

При реализации этого метода необходимо выполнить $n+l-1$ умножений при вычислениях по (4.15). При вычислении (4.16) умножения возникают при умножении значений $C(\gamma_i)$ на коэффициенты интерполяционных полиномов. Как видно из (4.17) эти коэффициенты могут быть и обыкновенными дробями.

Что касается числителей этих дробей, то при выборе значений γ_i небольшими целыми числами они тоже будут небольшими целыми числами. Если их значения являются степенями 2, то можно воспользоваться короткими операциями сдвига. В противных случаях можно заменить умножение на целое число несколькими сложениями. Однако, при увеличении длин сворачиваемых последовательностей количество сложений будет быстро возрастать, что является недостатком этого алгоритма.

Наличие знаменателей требует выполнения операций вещественного деления. Избавиться от них можно с помощью приема переноса их в постоянную сворачиваемую последовательность. Если в рассматриваемой задаче последовательность $\bar{b} = \{b[0], b[1], ..., b[l-1]\}$ и соответствующий ей полином $B(x) = \sum_{i=0}^{l-1} b[i] x^i$ постоянные (ядро свертки), то в (4.15) вместо значений полинома $B(\gamma_i)$ достаточно применить

$$\overline{B}(\gamma_i) = \frac{B(\gamma_i)}{\prod_{j=0\ j\neq i}^{l+n-2} (\gamma_i-\gamma_j)}. \tag{4.18}$$

Рассмотрим конкретный пример. Пусть необходимо синтезировать алгоритм вертки двух последовательностей длиной 3 каждая: $\bar{a} = \{a[0], a[1], a[2]\}$ и

$\bar{b} = \{b[0], b[1], b[2]\}$. Соответствующие им полиномы $A(x)$ и $B(x)$ будут иметь степень 2:

$$A(x) = a[0] + a[1] \cdot x + a[2] \cdot x^2; \quad B(x) = b[0] + b[1] \cdot x + b[2] \cdot x^2. \qquad (4.19)$$

Свертка $\bar{c}$ последовательностей $\bar{a}$ и $\bar{b}$ будет иметь длину 5:

$$\bar{c} = \bar{a} * \bar{b} = \{c[0], c[1], c[2], c[3], c[4]\}, \qquad (4.20)$$

а соответствующий ей полином будет иметь степень 4:

$$C(x) = c[0] + c[1] \cdot x + c[2] \cdot x^2 + c[3] \cdot x^3 + c[4] \cdot x^4. \qquad (4.21)$$

Для интерполяции $C(x)$ необходимо знать его значения в 5 точках. Пусть это будут следующие значения:

$$\gamma_0 = 0, \quad \gamma_1 = +1, \quad \gamma_2 = -1, \quad \gamma_3 = +2, \quad \gamma_4 = -2. \qquad (4.22)$$

Интерполяционные полиномы Лагранжа в соответствии с (4.17) и (4.22) будут иметь вид:

$$L_0(x) = \frac{(x-\gamma_1)(x-\gamma_2)(x-\gamma_3)(x-\gamma_4)}{(\gamma_0-\gamma_1)(\gamma_0-\gamma_2)(\gamma_0-\gamma_3)(\gamma_0-\gamma_4)} = \frac{1}{4}\left(x^4 - 5x^2 + 4\right).$$

Аналогично:

$$L_1(x) = -\frac{1}{6}\left(x^4 + x^3 - 4x^2 - 4x\right),$$

$$L_2(x) = -\frac{1}{6}\left(x^4 - x^3 - 4x^2 + 4x\right),$$

$$L_3(x) = \frac{1}{24}\left(x^4 + 2x^3 - x^2 - 2x\right),$$

$$L_4(x) = \frac{1}{24}\left(x^4 - 2x^3 - x^2 + 2x\right).$$

Находим значения полиномов $A(x)$ и $B(x)$ в точках (4.22), но при этом значения полинома $B(x)$ будем сразу умножать на коэффициенты интерполяционных полиномов Лагранжа:

$$\begin{aligned}
A(\gamma_0) &= a[0]; & B(\gamma_0) &= \frac{1}{4}b[0];\\
A(\gamma_1) &= a[0] + a[1] + a[2]; & B(\gamma_1) &= -\frac{1}{6}(b[0] + b[1] + b[2]);\\
A(\gamma_2) &= a[0] - a[1] + a[2]; & B(\gamma_2) &= -\frac{1}{6}(b[0] - b[1] + b[2]);\\
A(\gamma_3) &= a[0] + 2a[1] + 4a[2]; & B(\gamma_3) &= \frac{1}{24}(b[0] + 2b[1] + 4b[2]);\\
A(\gamma_4) &= a[0] - 2a[1] + 4a[2]; & B(\gamma_3) &= \frac{1}{24}(b[0] - 2b[1] + 4b[2]).
\end{aligned} \qquad (4.23)$$

Тогда интерполяционные полиномы Лагранжа примут вид:

$$L_0(x) = x^4 - 5x^2 + 4;$$
$$L_1(x) = x^4 + x^3 - 4x^2 - 4x;$$
$$L_2(x) = x^4 - x^3 - 4x^2 + 4x;$$
$$L_3(x) = x^4 + 2x^3 - x^2 - 2x;$$
$$L_4(x) = x^4 - 2x^3 - x^2 - 2x.$$

Если теперь по (4.15) и (4.23) вычислить значения полинома $C(x)$ в 5 точках, и эти значения подставить в (4.16), то после приведения подобных относительно переменной x получим восстановленный полином $C(x)$, коэффициенты которого и есть отсчеты линейной свертки последовательностей $\bar{a}$ и $\bar{b}$.

Как показывает анализ процесса вычислений линейной свертки в данном случае требуется всего 5 умножений и 25 сложений, тогда как непосредственными вычислениями требуется 9 умножений и 8 сложений.

4.2.2 Алгоритм Винограда вычисления коротких сверток

Алгоритм вычисления свертки по методу Винограда [37–39] основан, как и предыдущий метод Кука-Тоома, на представлении сворачиваемых последовательностей полиномами. Однако восстановление результирующего полинома-произведения производится с помощью его представления по полиномиальному модулю, который допускает разложение на полиномы низших степеней.

Пусть, как и в предыдущем подразделе, требуется построить произведение двух полиномов $A(x) = \sum_{i=0}^{n-1} a[i]x^i$ и $B(x) = \sum_{i=0}^{l-1} b[i]x^i$, степени которых $(n-1)$ и $(l-1)$ соответственно.

Рассмотрим полином $M(x)$ такой, что его степень выше суммы степеней полиномов $A(x)$ и $B(x)$:

$$deg(M(x)) > deg(A(x)) + deg(B(x)). \qquad (4.24)$$

В этом случае вполне очевидно, что

$$C(x) = A(x) \cdot B(x) = A(x) \cdot B(x) \ (mod \ M(x)). \qquad (4.25)$$

Но поскольку мы допустили, что полином $M(x)$ допускает разложение на полиномы низших степеней:

$$M(x) = \prod_{i=0}^{K-1} m^{(i)}(x), \qquad (4.26)$$

то задача построения $A(x) \cdot B(x) \ (mod \ M(x))$ декомпозируется в совокупность задач построения $A(x) \cdot B(x) \ (mod \ m^{(i)}(x))$. В том случае, когда полином $m^{(t)}(x)$ имеет линейный вид:

$$m^{(i)}(x) = x - \alpha_i, \qquad (4.27)$$

вычисление $A(x) \cdot B(x) \ (mod \ m^{(i)}(x))$ особенно простое. Так как остаток от произведения полиномов равен остатку от произведения остатков по тому же модулю и в соответствии с теоремой Безу имеем:

$$C^{(i)}(x)=(x)C(x)\ \left(mod\ m^{(i)}(x)\right)=$$
$$=\left(A(x)\cdot B(x)\ \left(mod\ m^{(i)}(x)\right)\right)\ \left(mod\ m^{(i)}(x)\right)=A(\alpha_i)\cdot B(\alpha_i). \quad (4.28)$$

Полином-произведение $C(x)$ может быть восстановлен по теореме, обратной Китайской теореме об остатках для полиномов:

$$C(x)=\sum_{i=0}^{K-1} f^{(i)}(x)\cdot A(\alpha_i)\cdot B(\alpha_i). \quad (4.29)$$

При такой схеме вычислений операция умножения вещественных чисел необходима при вычислении $A(\alpha_i)\cdot B(\alpha_i)$. Кроме того, умножения могут встретиться при вычислениях по формулам (4.29). Однако, если числа α_i небольшие по абсолютной величине целые числа, то каждое из умножения можно заменить $|\alpha_i|-1$ сложений. Для больших длин сворачиваемых последовательностей количество сложений резко возрастает, что является недостатком данной модификации алгоритма Винограда.

Рассмотрим пример построения линейной свертки с помощью алгоритма Винограда двух последовательностей, состоящих из трех отсчетов каждая:

$$A(x)=a[0]+a[1]\cdot x+a[2]\cdot x^2; \qquad B(x)=b[0]+b[1]\cdot x+b[2]\cdot x^2.$$

Так как $deg(A(x))+deg(B(x))=4$, то выберем $deg(M(x))=5$, а сам полином $M(x)$ в виде произведения линейных полиномов. Пусть

$$M(x)=x(x+1)(x-1)(x+2)(x-2)=x^5-5x^3+4x. \quad (4.30)$$

Вычисляем с помощью теоремы Безу остатки по всем модулям полиномов $A(x)$ и $B(x)$:

$$\begin{aligned}
A^{(0)}(x)&=A(x)(mod\ x)=a[0];\\
A^{(1)}(x)&=A(x)(mod\ (x+1))=a[0]-a[1]+a[2];\\
A^{(2)}(x)&=A(x)(mod\ (x-1))=a[0]+a[1]+a[2];\\
A^{(3)}(x)&=A(x)(mod\ (x+2))=a[0]-2a[1]+4a[2];\\
A^{(4)}(x)&=A(x)(mod\ (x-2))=a[0]+2a[1]+4a[2].
\end{aligned} \quad (4.31)$$

Так как остатки от полинома $B(x)$ имеют точно такую же структуру, то здесь не будем их приводить.

Как видно из структуры формул (4.31), здесь есть умножение только на небольшие целые коэффициенты. Эти умножения можно выполнить либо короткими операциями сдвига регистра, либо заменить сложениями. Например,

$$a[0]+2a[1]+4a[2]=a[0]+a[1]+a[1]+a[2]+a[2]+a[2]+a[2].$$

После этого по (4.28) можно определить остатки $C^{(i)}(x)$, $i=0,...,5$ и восстановить полином $C(x)$ по теореме, обратной Китайской теореме об остатках для полиномов с помощью соотношения

$$n^{(i)}(x)\cdot M^{(i)}(x)+N^{(i)}(x)\cdot\prod_{\substack{k=0\\k\neq i}}^{K} M^{(k)}(x)=1, \quad (4.32)$$

где $n^{(i)}(x)$ и $N^{(i)}(x)$ – полиномы соответствующих степеней, которые могут быть найдены методом неопределенных коэффициентов.

Пусть $i=0$. Тогда $M^{(0)}(x)=x$, $\prod\limits_{\substack{k=0\\k\neq i}}^{K} M^{(k)}(x)=x^4-5x^2+4$. Отсюда следует: $deg\left(n^{(0)}(x)\right)=3$, $deg\left(N^{(0)}(x)\right)=0$. Поэтому можно положить:

$$n^{(0)}(x)=\alpha_3x^3+\alpha_2x^2+\alpha_1x+\alpha_0, \qquad N^{(0)}(x)=\beta.$$

Подставляя в (4.32), методом неопределенных коэффициентов можно найти:

$$N^{(0)}(x)=\frac{1}{4}.$$

Действуя аналогичным образом, получим:

$$N^{(1)}(x)=\frac{1}{6};\quad N^{(2)}(x)=\frac{1}{6};\quad N^{(3)}(x)=-\frac{1}{24};\quad N^{(4)}(x)=\frac{1}{24}.$$

Так как $f^{(i)}(x)=N^{(i)}(x)\cdot\prod\limits_{\substack{k=0\\k\neq i}}^{K} M^{(k)}(x)$, то полином-произведение $C(x)$ можно вычислить по (4.29):

$$\begin{aligned}C(x)=&\frac{1}{4}\left(x^4-5x^2+4\right)\cdot C^{(0)}(x)-\frac{1}{6}\left(x^4-x^3-4x^2+4x\right)\cdot C^{(1)}(x)+\\&+\frac{1}{6}\left(x^4+x^3-4x^2-4x\right)\cdot C^{(2)}(x)+\frac{1}{24}\left(x^4-2x^3-x^2+2x\right)\cdot C^{(3)}(x)-\\&-\frac{1}{24}\left(x^4+2x^3-x^2-2x\right)\cdot C^{(4)}(x).\end{aligned} \qquad (4.33)$$

Группируя члены по одинаковым степеням x, получим полином $C(x)$, коэффициенты которого и есть отсчеты линейной свертки (4.12):

$$\begin{aligned}c[0]&=\frac{1}{4}C^{(0)}(x);\\c[1]&=-\frac{2}{3}C^{(1)}(x)+\frac{2}{3}C^{(2)}(x)+\frac{1}{12}C^{(3)}(x)+\frac{1}{12}C^{(4)}(x);\\c[2]&=-\frac{5}{4}C^{(0)}(x)+\frac{2}{3}C^{(1)}(x)-\frac{2}{3}C^{(2)}(x)-\frac{1}{24}C^{(3)}(x)+\frac{1}{24}C^{(4)}(x);\\c[3]&=\frac{1}{6}C^{(1)}(x)+\frac{1}{6}C^{(2)}(x)-\frac{1}{12}C^{(3)}(x)-\frac{1}{12}C^{(4)}(x);\\c[4]&=\frac{1}{4}C^{(0)}(x)-\frac{1}{6}C^{(1)}(x)+\frac{1}{6}C^{(2)}(x)+\frac{1}{24}C^{(3)}(x)-\frac{1}{24}C^{(4)}(x).\end{aligned} \qquad (4.34)$$

Все умножения в (4.34) на дробные коэффициенты можно выполнить предварительно с отсчетами ядра свертки $\bar{b}=\{b[0],b[1],b[2],b[3],b[4]\}$. С учетом этого данный алгоритм включает всего 5 умножений и 21 сложение.

4.2.3 Применение преобразования Фурье

Для вычисления циклической свертки преобразование Фурье интересно тем, что выполняется теорема о свертке: преобразование Фурье свертки сворачиваемых последовательностей равно произведению преобразований Фурье сворачиваемых последовательностей.

Дискретное преобразование Фурье сопоставляет последовательности вещественных или комплексных отсчетов $\bar{a} = \{a[0], ..., a[n-1]\}$ последовательность комплексных отсчетов $\bar{A} = \{A[0], ..., A[n-1]\}$, определяемых правилом:

$$A[i] = \sum_{k=0}^{n-1} \omega^{ki} a[k], \qquad i = 0,...,n-1, \tag{4.35}$$

где: $\omega = e^{j\frac{2\pi}{n}}$, $j^2 = -1$.

В дальнейшем преобразование Фурье будем обозначать буквой F:

$$\bar{A} = F(\bar{a}). \tag{4.36}$$

Существует и обратное преобразование Фурье – F^{-1}. Из (4.35) следует:

$$a[k] = \frac{1}{n} \sum_{i=0}^{n-1} \omega^{-ik} A[i] \quad \text{или} \quad \bar{a} = F^{-1}(\bar{A}). \tag{4.37}$$

В соответствии с теоремой о свертке, если $\bar{c} = \bar{a} * \bar{b}$, то:

$$C[i] = A[i] \cdot B[i]. \tag{4.38}$$

Таким образом, чтобы найти циклическую свертку двух последовательностей, необходимо найти преобразования Фурье (4.35) сворачиваемых последовательностей, покомпонентно их перемножить по (4.38), и подвергнуть полученную последовательность обратному преобразованию Фурье (4.37):

$$\bar{a} * \bar{b} = F^{-1}\left(F(\bar{a}) \cdot F(\bar{b})\right). \tag{4.39}$$

Как видим, основная идея применения преобразования Фурье к расчету циклической свертки также заключается в переходе в другое пространство, в котором расчеты выполняются с меньшими вычислительными затратами.

Для того, чтобы вычисление циклической свертки по схеме (4.39) было эффективным, необходимо, чтобы прямое и обратное преобразования Фурье не требовали большого объема вычислений. Однако преобразование Фурье само по себе является сверткой. Здесь следует отметить, что для эффективного выполнения преобразования Фурье разработаны различные алгоритмы, которые называются быстрым преобразованием Фурье (БПФ). В основе алгоритмов БПФ лежат методы декомпозиции исходной задачи в совокупность задач малой размерности. Именно такой подход лежит в основе алгоритмов Кули-Тьюки и Гуда-Томаса, основные черты которых будут рассмотрены в следующих разделах, следуя работе Р. Блейхута [37].

4.2.4 Алгоритм Кули-Тьюки

Пусть в преобразовании Фурье

$$A[i]=\sum_{k=0}^{n-1}\omega^{ki}a[k]$$

длина n преобразуемой последовательности $\bar{a}=\{a[0],...,a[n-1]\}$ есть число составное:

$$n=m\cdot l\,. \tag{4.40}$$

Тогда любой индекс членов последовательности $\bar{a}$ можно представить в виде чисел по основанию m, а индекс членов последовательности $\bar{A}$ – в виде чисел по основанию l:

$$k=k'+m\cdot k'',\ \ k'=0,...,m-1,\ \ k''=0,...,l-1, \tag{4.41}$$

$$i=i''+l\cdot i',\ \ i'=0,...,m-1,\ \ i''=0,...,l-1. \tag{4.42}$$

Подставляем эти числа в дискретное преобразование Фурье:

$$A[i''+l\cdot i']=\sum_{k=0}^{n-1}\omega^{(k'+m\cdot k'')(i''+l\cdot i')}a[k'+m\cdot k'']. \tag{4.43}$$

Если в показателе степени раскрыть скобки, то получим:

$$\omega^{mlk''i'}=\omega^{nk''i'}=\left(\omega^{j\cdot 2\pi}\right)^{k''i'}=1. \tag{4.44}$$

Кроме того, введем обозначения:

$$\omega^m=\gamma,\ \omega^l=\beta, \tag{4.45}$$

а также одномерные переменные запишем в виде двумерных по правилам

$$a[k'+m\cdot k'']=a[k',k''],\qquad A[i''+l\cdot i']=A[i',i'']. \tag{4.46}$$

Выражение (4.43) с учетом (4.44) – (4.46) примет вид:

$$A[i',i'']=\sum_{k'=0}^{m-1}\beta^{k'i'}\left(\omega^{k'i''}\sum_{k''=0}^{l-1}\gamma^{k''i''}a[k',k'']\right). \tag{4.47}$$

Выражение (4.47) имеет структуру двухуровневого преобразования Фурье (именно двухуровневого, а не двумерного, так как присутствует множитель «регулировки» $\omega^{k'i''}$). Мнемонически его можно представить так:

$$F\left(A[i',i'']\right)=F\left(\omega^{k'i''}F\left(a[k',k'']\right)\right). \tag{4.48}$$

Таким образом начальная задача n-точечного преобразования Фурье сводится к m задачам l-точечного преобразования Фурье и l задачам m-точечного преобразования Фурье. Этим алгоритм Кули-Тьюки напоминает метод дискретного динамического программирования [47, 48], где задача многомерной оптимизации сводится к совокупности задач меньшей размерности, благодаря чему значительно уменьшается объем вычислений.

Как показано в [37], полное число умножений (комплексных), при реализации алгоритма Кули-Тьюки определяется формулой:

$$M=n(m+l+1).$$

Это значительно меньше, чем при вычислениях по формуле преобразования Фурье (4.39), где требуется n^2 умножений.

Здесь же необходимо отметить, что, поскольку исходная задача разбивается на множество задач меньших размерностей, то очень важным является разработка эффективных алгоритмов вычисления свертки для последовательностей небольших длин.

4.2.5 Алгоритм Гуда-Томаса

Пусть в преобразовании Фурье

$$A[k]=\sum_{i=0}^{n-1}\omega^{ki}a[i] \tag{4.49}$$

длина n преобразуемой последовательности $\bar{a}=\{a[0],...,a[n-1]\}$ есть число не просто составное, а состоящее из двух взаимно простых делителей n' и n'':

$$n=n'\cdot n''. \tag{4.50}$$

Алгоритм Гуда-Томаса основан на Китайской теореме об остатках для целых чисел. Одномерные индексы входной последовательности i превратим в двумерные по правилам:

$$i'=i(mod\ n');\qquad i''=i\,(mod\ n''). \tag{4.51}$$

В соответствии с Китайской теореме об остатках существуют такие целые числа N' и N'', что выполняется равенство

$$i=i'N''n''+i''N'n'\,(mod\ n'), \tag{4.52}$$

где

$$N'n'+N''n''=1. \tag{4.53}$$

Индексы выходной последовательности k превращаются в двумерные по правилам:

$$k'=N''k\ \ (mod\ n');\qquad k''=N'k\ \ (mod\ n'') \tag{4.54}$$

и

$$k=n''k'+n'k''\,(mod\ n). \tag{4.55}$$

Выражение (4.49) с учетом (4.52) и (4.55) принимает вид:

$$A[n''k'+n'k'']=\sum_{i''=0}^{n''-1}\sum_{i'=0}^{n'-1}\omega^{(i'N''n''+i''N'n')(n''k'+n'k'')}a[i'N''n''+i''N'n']. \tag{4.56}$$

Если учесть, что

$$\omega^{n'n''}=\omega^{n}=1$$

и ввести обозначения:

$$\beta=\omega^{N''(n'')^2};\qquad \gamma=\omega^{N'(n')^2},$$

то выражение (4.56) при изображении одномерных индексов двумерными примет вид:

$$A[k',k'']=\sum_{i''=0}^{n''-1}\sum_{i'=0}^{n'-1}\beta^{i'k'}\gamma^{i''k''}a[i',i'']. \tag{4.57}$$

Это двумерное преобразование Фурье, для реализации которого требуется несколько меньше умножений, чем в алгоритме Кули-Тьюки, а именно – $n(n'+n'')$.

Таким образом, здесь также реализована идея декомпозиции исходной задачи в совокупность задач меньших размерностей.

4.3 Линейная свертка с использованием ГЧС

Так как сложность вычисления линейной свертки последовательностей длиной n есть $O(n^2)$, и быстро увеличивается с ростом n, то используются методы «быстрых» вычислений. Как отмечалось ранее, одним из наиболее распространенных методов является свертка с использованием быстрого преобразования Фурье (БПФ) со сложностью $O(n\log n)$. В основе многочисленных алгоритмов БПФ лежит декомпозиция исходной задачи большой размерности в большое количество задач малой размерности. Поэтому весьма важным является разработка таких методов решения задач малой размерности, которые используют, возможно, меньшее число вещественных операций.

В соответствии с теоремой Винограда для линейной свертки [37, 38] нижняя граница количества вещественных умножений при выполнении линейной свертки последовательностей длины n равна $2n-1$, что может быть ориентиром при оценке качества алгоритма. Однако, как отмечается в работе [38] «Во многих случаях в качестве меры сложности вычислений берется количество арифметических операций в алгоритме. Хотя и существует грубое соответствие между общей и арифметической сложностью алгоритма, все же практическая ценность вычислительного метода зависит от многих факторов. Эффективность алгоритма определяется не только числом операций, но и такими параметрами, как число перемещений данных, стоимость вспомогательных операций, общая структурная сложность, различные возможности, представляемые используемой вычислительной системой, искусство программиста. Поэтому упорядочение алгоритмов по их действительной эффективности, выраженной временем выполнения, является весьма трудным делом, так что сравнения, основанные лишь на числе арифметических операций, должны быть «взвешены» с учетом факторов, возникающих при конкретных реализациях этих алгоритмов». Это говорит о том, что необходимы алгоритмы с самыми разнообразными свойствами. В данном разделе будут рассмотрены алгоритмы выполнения свертки, основанные на переходе к гиперкомплексным пространствам. Основы такого подхода разработаны авторами и изложены в работе [49].

Пусть необходимо выполнить линейную свертку двух числовых последовательностей длиной 2^n. Будем рассматривать эти числовые последовательности как компоненты гиперкомплексных чисел, принадлежащих некоторой ГЧС Γ_1 размерностью $\dim\Gamma_1 = 2^n$. Произведение этих чисел будет содержать парные произведения компонентов свертываемых числовых последовательностей. Однако они будут комбинироваться в суммы не в таком составе, как это нужно для организации компонентов свертки. Кроме того, число вещественных умножений при умножении гиперкомплексных чисел в общем случае равно 2^{2n}, то есть столько же, как и при прямом вычислении свертки, то есть, нет никакого выигрыша.

Таким образом, здесь возникают две проблемы: первая – это снижение количества вещественных операций при умножении гиперкомплексных чисел; вторая – организация выбора парных произведений компонентов свертки. Решение этих двух проблем позволяет синтезировать такие алгоритмы свертки, которые будут по количеству операций эффективнее других алгоритмов выполнения свертки.

Для решения первой проблемы можно перейти в такую ГЧС, изоморфную исходной, таблица умножения которой заполнена слабо. Такие пары ГЧС существуют и описаны в [1, 2], а также в подразделе 1.3.4.

Переход между такими ГЧС требует только выполнения операций сложения вещественных чисел.

Решение второй проблемы зависит от конкретного вида используемых ГЧС и будет рассмотрено ниже.

Как отмечалось ранее линейная свертка выполняется по формуле [37 – 39]

$$z[k] = \sum_{m=-\infty}^{k} x[m]y[k-m]. \qquad (4.58)$$

Если рассматривается линейная свертка двух последовательностей длиной по 2^n элементов, то при прямом вычислении ее $2^{n+1}-1$ отсчетов необходимо произвести 2^{2n} вещественных умножений. С помощью гиперкомплексных числовых систем можно уменьшить это количество.

Рассмотрим иллюстративный пример. Пусть $n=1$, то есть вычисляется линейная свертка двух числовых последовательностей – $\{x[0], x[1]\}$ и $\{y[0], y[1]\}$. В соответствии с (4.58) свертка будет иметь три отсчета:

$$\begin{aligned} z[-1] &= x[0]y[0] \\ z[0] &= x[0]y[1] + x[1]y[0]. \\ z[1] &= \quad\quad x[1]y[1] \end{aligned} \qquad (4.59)$$

Как видно из (4.59), для выполнения данной свертки необходимо выполнить 4 вещественных умножения.

Будем рассматривать элементы последовательностей $\{x[0], x[1]\}$ и $\{y[0], y[1]\}$ как компоненты гиперкомплексных чисел, принадлежащие ГЧС размерности 2 $W(e,2)$:

$$X = x[0]e_0 + x[1]e_1, \qquad (4.60)$$

$$Y = y[1]e_0 + y[0]e_1. \qquad (4.61)$$

Второе число записано в инверсном виде.

В соответствии с таблицей умножения ГЧС W:

$$XY = (x[0]y[1] + x[1]y[0])e_0 + (x[0]y[0] + x[1]y[1])e_1. \qquad (4.62)$$

Правая часть выражения (4.62) содержит все парные произведения, необходимые для формирования свертки (4.59). Более того, компонента при e_0 полностью совпадает с отсчетом $z[1]$, а компонента при e_1 равна сумме остальных отсчетов. Но количество вещественных умножений также равно 4, как и в

(4.59). Таким образом, возникают две проблемы: во-первых, уменьшить число умножений в (4.62); и, во-вторых, разделить отсчеты $z[1]$ и $z[2]$.

Для уменьшения количества умножений целесообразно перейти из системы W в систему W_1. Тогда для перевода чисел $X, Y \in W$ в числа $X_1, Y_1 \in W_1$ необходимо в (4.60) и (4.61) подставить левые преобразования (1.36). Тогда

$$X_1 = (x[0] + x[1]) f_0 + (x[0] - x[1]) f_1; \quad (4.63)$$

$$Y_1 = (y[1] + y[0]) f_0 + (y[1] - y[0]) f_1. \quad (4.64)$$

Такой переход требует дополнительно 4 сложения. Однако, учитывая, что обычно числовая последовательность сворачивается с постоянным ядром, переход элементов которого в W_1 сделан заранее, то можно считать, что переход из системы W в систему W_1 требует только 2 сложения.

Умножение чисел (4.63) и (4.64) в соответствии с таблицей умножения W_1:

$$X_1 Y_1 = \alpha_0 f_0 + \alpha_1 f_1 = (x[0] + x[1])(y[1] + y[0]) f_0 + (x[0] - x[1])(y[1] - y[0]) f_1 \quad (4.65)$$

требует 2 умножения.

Обратный переход из W_1 в систему W в соответствии с правым преобразованием (1.36) будет иметь вид:

$$XY = \frac{\alpha_0 + \alpha_1}{2} e_0 + \frac{\alpha_0 - \alpha_1}{2} e_1, \quad (4.66)$$

что также требует 2 сложения. Деление на 2 – это короткая операция, требующая только сдвига регистра. Поэтому ее можно не учитывать.

Приравнивая правые части (4.62) и (4.66), получим:

$$x[0]y[1] + x[1]y[0] = \frac{\alpha_0 + \alpha_1}{2}, \quad (4.67)$$

$$x[0]y[0] + x[1]y[1] = \frac{\alpha_0 - \alpha_1}{2}. \quad (4.68)$$

Если вычислить заранее одно из слагаемых в левой части (4.68), например, $x[0]y[0]$, то компоненты свертки (4.6) будут вычисляться по следующим формулам:

$$\begin{aligned} z[-1] &= x[0]y[0] \\ z[0] &= \frac{\alpha_0 + \alpha_1}{2} \\ z[1] &= \frac{\alpha_0 - \alpha_1}{2} - x[0]y[0] \end{aligned}, \quad (4.69)$$

что требует 3 умножений и 5 сложений. При расчетах по (4.68) требуется соответственно 4 умножения и 1 сложение. В данном случае эффект скорее отрицательный. Этот пример, как было отмечено выше, носит чисто иллюстративный характер для демонстрации идеи. Однако, как будет показано далее, при возрастании длины сворачиваемой последовательности эффект становится положительным.

Для наглядного отображения структуры вычислений построим в координатах $x0y$ парные произведения элементов обоих последовательностей. Значку $\otimes$ соответствует парное произведение, полученное в результате перемножения гиперкомплексных чисел, значку $\times$ – непосредственно вычисляемое произведение. Пустая клетка соответствует парному произведению, при вычислении которого не нужно умножение элементов последовательностей.

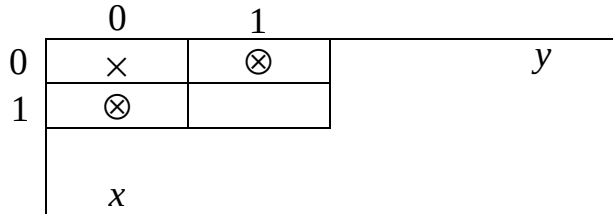


Рис. 4.1. Структура вычислительного процесса при $n=1$

Пусть $n=2$, то есть вычисляется линейная свертка двух числовых последовательностей – $\{x[0],x[1],x[2],x[3]\}$ и $\{y[3],y[2],y[1],y[0]\}$. В соответствии с (4.58) свертка будет иметь 7 отсчетов:

$$\begin{aligned}
z[-3] &= x[0]y[0] \\
z[-2] &= x[0]y[1]+x[1]y[0] \\
z[-1] &= x[0]y[2]+x[1]y[1]+x[2]y[0] \\
z[0] &= x[0]y[3]+x[1]y[2]+x[2]y[1]+x[3]y[0]. \\
z[1] &= x[1]y[3]+x[2]y[2]+x[3]y[1] \\
z[2] &= x[2]y[3]+x[3]y[2] \\
z[3] &= x[3]y[3]
\end{aligned} \qquad (4.70)$$

Для выполнения данной свертки необходимо выполнить 16 вещественных умножений и 9 сложений.

Будем рассматривать элементы последовательностей $\{x[0],x[1],x[2],x[3]\}$ и $\{y[3],y[2],y[1],y[0]\}$ как компоненты гиперкомплексных чисел ГЧС $W^{(2)}(e,4)[3]$:

$$X = x[0]e_0 + x[1]e_1 + x[2]e_2 + x[3]e_3: \qquad (4.71)$$

$$Y = y[3]e_0 + y[2]e_1 + y[1]e_2 + y[0]e_3. \qquad (4.72)$$

Тогда в соответствии с таблицей умножения ГЧС $W^{(2)}(e,4)$ произведение (4.71) и (4.72) имеет вид:

$$XY = \sum_{i=0}^{3} \alpha_i e_i , \qquad (4.73)$$

где компоненты α_i имеют вид:

$$\begin{aligned}
\alpha_0 &= x[0]y[3]+x[1]y[2]+x[2]y[1]+x[3]y[0];\\
\alpha_1 &= x[0]y[2]+x[1]y[3]+x[2]y[0]+x[3]y[1];\\
\alpha_2 &= x[0]y[1]+x[1]y[0]+x[2]y[3]+x[3]y[2];\\
\alpha_3 &= x[0]y[0]+x[1]y[1]+x[2]y[2]+x[3]y[3].
\end{aligned} \qquad (4.74)$$

Для уменьшения количества умножений целесообразно перейти из системы $W^{(2)}(e,4)$ в систему $W_1^{(2)}(f,4)$ [3]. Тогда для перевода гиперкомплексных чисел, принадлежащих $W^{(2)}(e,4)$, в гиперкомплексные числа, принадлежащие $W_1^{(2)}(f,4)$, необходимо в (4.71) и (4.72) подставить преобразования (1.50). Тогда

$$\begin{aligned}
X_1 = \sum_{i=0}^{3}\beta_i f_i = (x[0]+x[1]+x[2]+x[3])f_0 + (x[0]-x[1]+x[2]-x[3])f_1 + \\
+ (x[0]+x[1]-x[2]-x[3])f_2 + (x[0]-x[1]-x[2]+x[3])f_3.
\end{aligned} \qquad (4.75)$$

Аналогично и Y_1:

$$Y_1 = \sum_{i=0}^{3}\gamma_i f_i. \qquad (4.76)$$

Такой переход требует дополнительно 16 сложений. Однако, учитывая, что обычно числовая последовательность сворачивается с постоянным ядром, переход элементов которого в $W_1^{(2)}(f,4)$ сделан заранее, то можно считать, что переход из системы $W^{(2)}(e,4)$ в систему $W_1^{(2)}(f,4)$ требует только 8 сложений.

Умножение чисел (4.75) и (4.76) в соответствии с таблицей умножения $W_1^{(2)}(f,4)$ имеет вид

$$X_1Y_1 = \sum_{i=0}^{3}\beta_i\gamma_i f_i \qquad (4.77)$$

и требует всего лишь 4 умножения.

Обратный переход из $W_1^{(2)}(f,4)$ в систему $W^{(2)}(e,4)$ в соответствии с (1.51) будет иметь вид:

$$\begin{aligned}
XY = \sum_{i=0}^{3}\alpha_i e_i = \frac{1}{4}(\beta_0\gamma_0+\beta_1\gamma_1+\beta_2\gamma_2+\beta_3\gamma_3)e_0 + \frac{1}{4}(\beta_0\gamma_0-\beta_1\gamma_1+\beta_2\gamma_2-\beta_3\gamma_3)e_1 + \\
+\frac{1}{4}(\beta_0\gamma_0+\beta_1\gamma_1-\beta_2\gamma_2-\beta_3\gamma_3)e_2 + \frac{1}{4}(\beta_0\gamma_0-\beta_1\gamma_1-\beta_2\gamma_2+\beta_3\gamma_3)e_3,
\end{aligned} \qquad (4.78)$$

что требует 8 сложений. Деление на 4 – это короткая операция, требующая только сдвига регистра. Поэтому ее можно не учитывать.

Тогда вычисления отсчетов свертки с указанием количества умножений (M) и сложений (A) будут иметь вид:

$$
\begin{aligned}
&z[-3] = x[0]y[0] && -1M\\
&z[-2] = x[0]y[1] + x[1]y[0] && -2M + 1A\\
&z[-1] = x[0]y[2] + x[1]y[1] + x[2]y[0] && -3M + 2A\\
&z[0] = \alpha_0 && -4M\\
&z[1] = \alpha_1 - (x[0]y[2] + x[2]y[0]) + x[2]y[2] && -1\mathrm{M} + 2A\\
&z[2] = \alpha_2 - x[0]y[1] + x[1]y[0] && -1A\\
&z[3] = \alpha_3 - (x[0]y[0] + x[1]y[1] + x[2]y[2]) && -3A
\end{aligned}
\qquad (4.79)
$$

Итого 11 умножений и 25 сложений (здесь учтены также 16 сложений, необходимых для перехода из $W^{(2)}(e,4)$ в $W_1^{(2)}(f,4)$ и обратно).

Построим диаграмму структуры вычислительного процесса подобно предыдущему случаю.

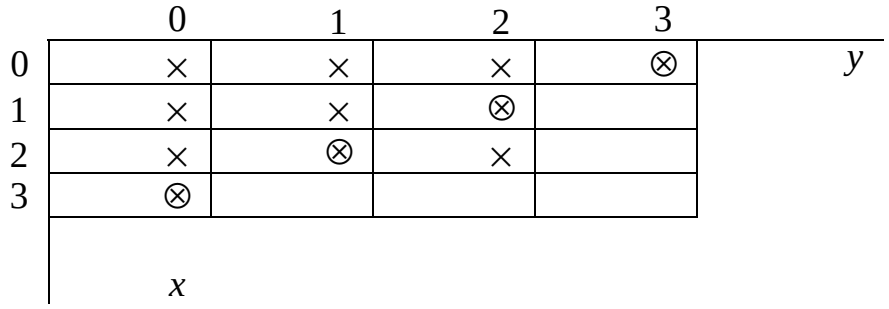


Рис. 4.2. Структура вычислительного процесса при $n = 2$

Рассмотрим случай $n = 3$, то есть вычисление линейной свертки двух числовых последовательностей – $\{x[0],...,x[7]\}$ и $\{y[7],...,y[0]\}$. В соответствии с (4.58) свертка будет иметь 15 отсчетов, выражения которых приведены ниже.

$$
\begin{aligned}
&z[-7] = x[0]y[0];\\
&z[-6] = x[0]y[1] + x[1]y[0];\\
&z[-5] = x[0]y[2] + x[1]y[1] + x[2]y[0];\\
&z[-4] = x[0]y[3] + x[1]y[2] + x[2]y[1] + x[3]y[0];\\
&z[-3] = x[0]y[4] + x[1]y[3] + x[2]y[2] + x[3]y[1] + x[4]y[0];\\
&z[-2] = x[0]y[5] + x[1]y[4] + x[2]y[3] + x[3]y[2] + x[4]y[1] + x[5]y[0];\\
&z[-1] = x[0]y[6] + x[1]y[5] + x[2]y[4] + x[3]y[3] + x[4]y[2] + x[5]y[1] + x[6]y[0];\\
&z[0] = x[0]y[7] + x[1]y[6] + x[2]y[5] + x[3]y[4] + x[4]y[3] + x[5]y[2] + x[6]y[1] + x[7]y[0]\\
&z[1] = x[1]y[7] + x[2]y[6] + x[3]y[5] + x[4]y[4] + x[5]y[3] + x[6]y[2] + x[7]y[1];\\
&z[2] = x[2]y[7] + x[3]y[6] + x[4]y[5] + x[5]y[4] + x[6]y[3] + x[7]y[2];\\
&z[3] = x[3]y[7] + x[4]y[6] + x[5]y[5] + x[6]y[4] + x[7]y[3];\\
&z[4] = x[4]y[7] + x[5]y[6] + x[6]y[5] + x[7]y[4];\\
&z[5] = x[5]y[7] + x[6]y[6] + x[5]y[7];\\
&z[6] = x[6]y[7] + x[7]y[6];\\
&z[7] = x[7]y[7].
\end{aligned}
\qquad (4.80)
$$

Для выполнения данной свертки необходимо выполнить 64 вещественных умножения и 49 сложений.

Как и в предыдущих случаях $(n=1,2)$, можно воспользоваться изоморфными ГЧС 8-ой размерности $W^{(3)}(e,8) \simeq W_1^{(3)}(f,8)$, оператор изоморфизма которых имеет вид [2]:

$$\begin{aligned}
e_0 &= f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7;\\
e_1 &= f_0 - f_1 + f_2 - f_3 + f_4 - f_5 + f_6 - f_7;\\
e_2 &= f_0 + f_1 - f_2 - f_3 + f_4 + f_5 - f_6 - f_7;\\
e_3 &= f_0 - f_1 - f_2 + f_3 + f_4 - f_5 - f_6 + f_7;\\
e_4 &= f_0 + f_1 + f_2 + f_3 - f_4 - f_5 - f_6 - f_7;\\
e_5 &= f_0 - f_1 + f_2 - f_3 - f_4 + f_5 - f_6 + f_7;\\
e_6 &= f_0 + f_1 - f_2 - f_3 - f_4 - f_5 + f_6 + f_7;\\
e_7 &= f_0 - f_1 - f_2 + f_3 - f_4 + f_5 + f_6 - f_7;
\end{aligned} \tag{4.81}$$

$$\begin{aligned}
f_0 &= (e_0 + e_1 + e_2 + e_3 + e_4 + e_5 + e_6 + e_7)/8;\\
f_1 &= (e_0 - e_1 + e_2 - e_3 + e_4 - e_5 + e_6 - e_7)/8;\\
f_2 &= (e_0 + e_1 - e_2 - e_3 + e_4 + e_5 - e_6 - e_7)/8;\\
f_3 &= (e_0 - e_1 - e_2 + e_3 + e_4 - e_5 - e_6 + e_7)/8;\\
f_4 &= (e_0 + e_1 + e_2 + e_3 - e_4 - e_5 - e_6 - e_7)/8;\\
f_5 &= (e_0 - e_1 + e_2 - e_3 - e_4 + e_5 - e_6 + e_7)/8;\\
f_6 &= (e_0 + e_1 - e_2 - e_3 - e_4 - e_5 + e_6 + e_7)/8;\\
f_7 &= (e_0 - e_1 - e_2 + e_3 - e_4 + e_5 + e_6 - e_7)/8.
\end{aligned} \tag{4.82}$$

Далее:

$$X = \sum_{i=0}^{7} x[i] e_i, \tag{4.83}$$

$$Y = \sum_{i=0}^{7} y[7-i] e_i. \tag{4.84}$$

Тогда в соответствии с таблицей умножения ГЧС $W^{(3)}(e,8)$ произведение XY имеет вид

$$XY = \sum_{i=0}^{7} \alpha_i e_i,$$

где компоненты α_i таковы:

$$\alpha_0 = x[0]y[7]+x[1]y[6]+x[2]y[5]+x[3]y[4]+x[4]y[3]+x[5]y[2]+x[6]y[1]+x[7]y[0];$$
$$\alpha_1 = x[0]y[6]+x[1]y[7]+x[2]y[4]+x[3]y[5]+x[4]y[2]+x[5]y[3]+x[6]y[0]+x[7]y[1];$$
$$\alpha_2 = x[0]y[5]+x[1]y[4]+x[2]y[7]+x[3]y[6]+x[4]y[1]+x[5]y[0]+x[6]y[3]+x[7]y[2];$$
$$\alpha_3 = x[0]y[4]+x[1]y[5]+x[2]y[6]+x[3]y[7]+x[4]y[0]+x[5]y[1]+x[6]y[2]+x[7]y[3];$$
$$\alpha_4 = x[0]y[3]+x[1]y[2]+x[2]y[1]+x[3]y[0]+x[4]y[7]+x[5]y[6]+x[6]y[5]+x[7]y[4];$$
$$\alpha_5 = x[0]y[2]+x[1]y[3]+x[2]y[0]+x[3]y[1]+x[4]y[6]+x[5]y[7]+x[6]y[4]+x[7]y[5];$$
$$\alpha_6 = x[0]y[1]+x[1]y[0]+x[2]y[3]+x[3]y[2]+x[4]y[5]+x[5]y[4]+x[6]y[7]+x[7]y[6];$$
$$\alpha_7 = x[0]y[0]+x[1]y[1]+x[2]y[2]+x[3]y[3]+x[4]y[4]+x[5]y[5]+x[6]y[6]+x[7]y[7].$$

Для перевода чисел из $W^{(3)}(e,8)$ в числа из ГЧС $W_1^{(3)}(f,8)$ необходимо в (4.83) и (4.84) подставить преобразования (4.81). Тогда

$$\begin{aligned} X_1 = \sum_{i=0}^{7} \beta_i f_i = {} & (x[0]+x[1]+x[2]+x[3]+x[4]+x[5]+x[6]+x[7])f_0 + \\ & + (x[0]-x[1]+x[2]-x[3]+x[4]-x[5]+x[6]-x[7])f_1 + \\ & + (x[0]+x[1]-x[2]-x[3]+x[4]+x[5]-x[6]-x[7])f_2 + \\ & + (x[0]-x[1]-x[2]+x[3]+x[4]-x[5]-x[6]+x[7])f_3 + \\ & + (x[0]+x[1]+x[2]+x[3]-x[4]-x[5]-x[6]-x[7])f_4 + \\ & + (x[0]-x[1]+x[2]-x[3]-x[4]+x[5]-x[6]+x[7])f_5 + \\ & + (x[0]+x[1]-x[2]-x[3]-x[4]-x[5]+x[6]+x[7])f_6 + \\ & + (x[0]-x[1]-x[2]+x[3]-x[4]+x[5]+x[6]-x[7])f_7 . \end{aligned} \qquad (4.85)$$

Аналогично и Y_1:

$$Y_1 = \sum_{i=0}^{7} \gamma_i f_i . \qquad (4.86)$$

Такой переход требует дополнительно 32 сложения.

Умножение чисел (4.84) и (4.85) в соответствии с таблицей умножения $W_1^{(3)}(f,8)$ имеет вид

$$X_1 Y_1 = \sum_{i=0}^{7} \beta_i \gamma_i f_i . \qquad (4.87)$$

Оно требует всего лишь 8 умножений.

Обратный переход из $W_1^{(3)}(f,8)$ в систему $W^{(3)}(e,8)$ осуществляется в соответствии с преобразованием (4.81). Он требует 32 сложения. Деление на 8 – это короткая операция, требующая только сдвига регистра. Поэтому ее можно не учитывать.

Тогда вычисления отсчетов свертки с указанием количества умножений (M) и сложений (A) будут иметь вид:

$z[-7] = x[0]y[0]$ $\qquad 1M$

$z[-6] = x[0]y[1]+x[1]y[0]$ $\qquad 2M+1A$

$z[-5] = x[0]y[2]+x[1]y[1]+x[2]y[0]$ $\qquad 3M+2A$

$$z[-4] = x[0]y[3] + x[1]y[2] + x[2]y[1] + x[3]y[0] \qquad 4M + 3A$$

$$z[-3] = x[0]y[4] + x[1]y[3] + x[2]y[2] + x[3]y[1] + x[4]y[0] \qquad 5M + 4A$$

$$z[-2] = x[0]y[5] + x[1]y[4] + x[2]y[3] + x[3]y[2] + x[4]y[1] + x[5]y[0] \qquad 6M + 5A$$

$$z[-1] = x[0]y[6] + x[1]y[5] + x[2]y[4] + x[3]y[3] + x[4]y[2] + x[5]y[1] + x[6]y[0] \qquad 7M + 6A$$

$$z[0] = \alpha_0 \qquad 8M + 7A$$

$$z[1] = \alpha_1 - (x[0]y[6] + x[2]y[4] + x[4]y[2] + x[6]y[0]) + x[2]y[6] + x[4]y[4] + x[6]y[2] \qquad 3M + 4A$$

$$z[2] = \alpha_2 - (x[0]y[5] + x[1]y[4] + x[5]y[0]) + x[4]y[5] + x[5]y[4] \qquad 2M + 3A$$

$$z[3] = \alpha_3 - \begin{pmatrix} x[0]y[4] + x[1]y[5] + x[2]y[6] + x[4]y[0] + x[4]y[6] + \\ + x[6]y[4] + x[6]y[2] \end{pmatrix} + \qquad 3M + 8A$$

$$+ x[4]y[6] + x[5]y[5] + x[6]y[4]$$

$$z[4] = \alpha_4 - (x[0]y[3] + x[1]y[2] + x[2]y[1] + x[3]y[0]) \qquad 1A$$

$$z[5] = \alpha_5 - (x[0]y[2] + x[1]y[3] + x[2]y[0] + x[3]y[1] + x[4]y[6] + x[6]y[4]) + x[6]y[6] \qquad 1M + 4A$$

$$z[6] = \alpha_6 - (x[0]y[1] + x[1]y[0] + x[2]y[3] + x[3]y[2] + x[4]y[5] + x[5]y[4]) \qquad 3A$$

$$z[7] = \qquad 1A$$

$$\alpha_7 - (x[0]y[0] + x[1]y[1] + x[2]y[2] + x[3]y[3] + x[4]y[4] + x[5]y[5] + x[6]y[6])$$

Рис. 4.3. Вычисления отсчетов свертки при $n = 3$

Можно подсчитать, что в этом случае требуется 45 умножений и 118 сложений (здесь учтены также 64 сложения, необходимых для перехода из $W_1^{(3)}(f,8)$ в $W^{(3)}(e,8)$ и обратно).

Построим диаграмму структуры вычислительного процесса

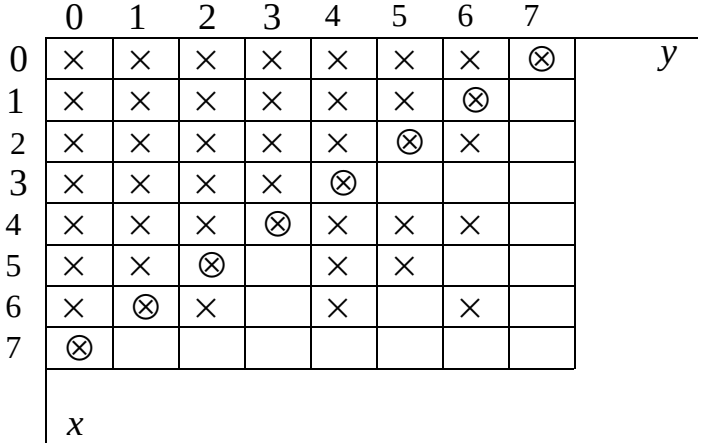


Рис. 4.4. Структура вычислительного процесса при $n = 3$

Совершенно аналогично могут быть построены вычислительные процессы и для более длинных сворачиваемых последовательностей отсчетов $(n>3)$, так как необходимые для этого ГЧС существуют [2].

Как было отмечено выше при обратных переходах $W_1^{(i)} \Rightarrow W^{(i)}$ необходимо компоненты чисел делить на 2^i. Это короткая операция, но даже ее можно устранить, поделив заранее постоянные для данной свертки компоненты последовательности Y на 2^i. В работах [37, 38] этот прием называется «спрятать двойку в ядро».

Из вышеприведенных оценок следует, что предложенный метод позволяет при вычислении линейной свертки значительно снизить количество умножений, как это показано в таблице 1.

Таблица 1

Зависимость количества умножений от длины последовательностей

№	Длина последовательностей a	Полное число умножений	Снижение числа умножений по алгоритму	То же в %
1	2	4	1	25
2	4	16	5	31.3
3	8	64	19	30

Как видно из таблицы, по мере увеличения длины последовательностей эффективность метода снижается, что требует дальнейшего повышения эффективности алгоритма.

4.4 Декомпозиционные алгоритмы вычисления линейной свертки

При линейной свертке последовательностей длиной 2 (то есть случай $n=1$) число умножений уменьшить нельзя, как это следует из теоремы Винограда для линейной свертки [37, 38]. Дальнейшее уменьшение числа умножений при вычислении линейной свертки последовательностей большей длины может быть достигнуто с помощью методов полной или частичной декомпозиции задачи.

4.4.1 Метод полной декомпозиции задачи

Полная декомпозиция задачи свертки последовательностей длиной 2^n отсчетов заключается в разложении её на 2^m задач свертки последовательностей длиной 2^{n-m} отсчетов.

Так, например, свертку последовательностей $\{x[0], x[1], x[2], x[3]\}$ и $\{y[0], y[1], y[2], y[3]\}$ можно свести к 4-м сверткам последовательностей $\{x[0], x[1]\}$ и $\{y[0], y[1]\}$, $\{x[0], x[1]\}$ и $\{y[2], y[3]\}$, $\{x[2], x[3]\}$ и $\{y[0], y[1]\}$, $\{x[2], x[3]\}$ и $\{y[2], y[3]\}$, как это показано на схеме вычислительного процесса рис. 4.5.

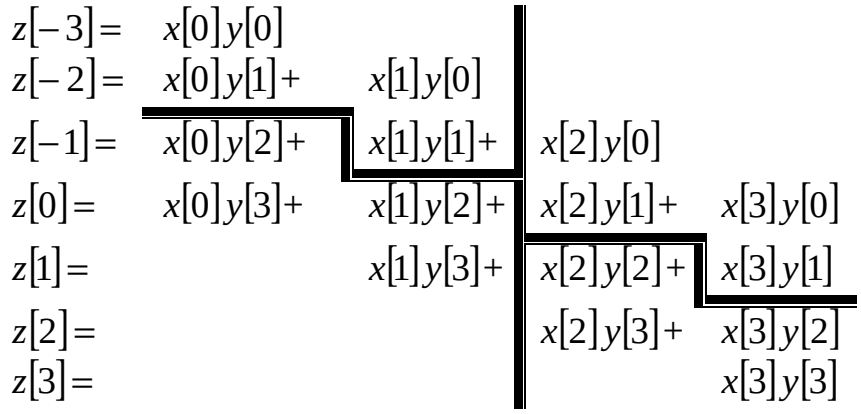


Рис. 4.5 Полная декомпозиция процесса вычисления линейной свертки

При этом число умножений будет $3 \cdot 4 = 12$, что лучше непосредственного вычисления свертки, но хуже алгоритма свертки последовательностей 4×4, показанного на рис 4.2. Кроме того, добавится 3 дополнительных сложений для «стыковки» трех отсчетов свертки (см. рис. 4.4).

Достоинством этого метода является возможность параллельного вычисления четырех сверток последовательностей $\{x[0], x[1]\}$ и $\{y[0], y[1]\}$, $\{x[0], x[1]\}$ и $\{y[2], y[3]\}$, $\{x[2], x[3]\}$ и $\{y[0], y[1]\}$, $\{x[2], x[3]\}$ и $\{y[2], y[3]\}$, что может сократить время решения всей задачи.

Как видно из вышеизложенного, вычисление сверток последовательностей длины 2^n отсчетов можно свести к вычислению 2^{n-1} сверток последовательностей 2×2 с последующей «стыковкой», на что уйдет некоторое количество сложений.

4.4.2 Метод частичной декомпозиции задачи

Этот метод заключается в том, что до перехода из ГЧС $W^{(n)}\left(e, 2^n\right)$ в систему $W_1^{(n)}\left(f, 2^n\right)$ производится $(n-1)$-а свертка последовательностей $\{x[0], x[i]\}$ и $\{y[0], y[i]\}$, $i = 1,2,..., n-1$. Полученные при этом значения произведений отсчетов и их парные суммы используются на дальнейших этапах расчета.

Рассмотрим случай $n = 2$. Здесь целесообразно сначала сделать свертку последовательностей $\{x[0], x[1]\}$ и $\{y[0], y[1]\}$. На это уйдет 3 умножения, и будут получены значения $x[1]y[1]$ и $x[0]y[1] + x[1]y[0]$. При этом полученное при преобразовании значение $x[0] + x[1]$ может быть использовано в дальнейшем, сокращая количество сложений. Далее выполняется свертка последовательностей $\{x[0], x[2]\}$ и $\{y[0], y[2]\}$. На это уже необходимо только 2 умножения и будут получены значения $x[2]y[2]$ и $x[0]y[2] + x[2]y[0]$, которые используются в дальнейшем. Далее вычислительный процесс строится также, как и ранее.

Тогда количество вычислений отсчетов свертки с указанием количества умножений будут иметь вид:

$$
\begin{array}{ll}
x[0]y[0], x[1]y[1], x[0]y[1]+x[1]y[0] & \\
x[2]y[2], x[0]y[2]+x[2]y[0] & \begin{array}{l}3\\2\end{array}\\
z[-3]=x[0]y[0] & -\\
z[-2]=x[0]y[1]+x[1]y[0] & -\\
z[-1]=x[0]y[2]+x[1]y[1]+x[2]y[0] & -\\
z[0]=\alpha_0 & 4\\
z[1]=\alpha_1-(x[0]y[2]+x[2]y[0])+x[2]y[2] & -\\
z[2]=\alpha_2-x[0]y[1]+x[1]y[0] & -\\
z[3]=\alpha_3-(x[0]y[0]+x[1]y[1]+x[2]y[2]) & -
\end{array}
\qquad (4.88)
$$

Несложно рассчитать, что, всего необходимо выполнить 9 умножений и 30 сложений (здесь учтены, также 5 умножений и 10 сложений для сверток 2×2, а также 14 сложений, необходимых для перехода из $W^{(2)}(e,4)$ в систему $W_1^{(2)}(f,4)$ и обратно).

Рассмотрим случай $n=3$. Здесь целесообразно сначала вычислить парное произведение $x[0]y[0]$, затем выполнить свертки последовательностей $\{x[0],x[1]\}$ и $\{y[0],y[1]\}$, $\{x[0],x[2]\}$ и $\{y[0],y[2]\},\ldots,\{x[0],x[6]\}$ и $\{y[0],y[6]\}$, на что необходимо затратить 13 умножений и 24 сложения. После этого производится перевод числа X из ГЧС $W^{(3)}(e,8)$ в систему $W_1^{(3)}(f,8)$, на что требуется 30 сложений. Умножение в $W_1^{(3)}(f,8)$ требует 8 вещественных умножений. Обратный переход из $W_1^{(3)}(f,8)$ в систему $W^{(3)}(e,8)$ требует 32 сложения. Далее вычислительный процесс строится также, как и в предыдущем случае, но количество операций изменится в меньшую сторону (в таблице указано только количество умножений):

$x[0]y[0], x[1]y[1], x[0]y[1]+x[1]y[0]$	3
$x[2]y[2], x[0]y[2]+x[2]y[0]$	2
$x[3]y[3], x[0]y[3]+x[3]y[0]$	2
$x[4]y[4], x[0]y[4]+x[4]y[0]$	2
$x[5]y[5], x[0]y[5]+x[5]y[0]$	2
$x[6]y[6], x[0]y[6]+x[6]y[0]$	2
$z[-7]=x[0]y[0]$	–
$z[-6]=x[0]y[1]+x[1]y[0]$	–
$z[-5]=x[1]y[1]+x[0]y[2]+x[2]y[0]$	–

$$z[-4] = x[0]y[3] + x[3]y[0] + x[1]y[2] + x[2]y[1] \quad 2$$

$$z[-3] = x[0]y[4] + x[4]y[0] + x[2]y[2] + x[1]y[3] + x[3]y[1] \quad 2$$

$$z[-2] = x[0]y[5] + x[5]y[0] + x[1]y[4] + x[4]y[1] + x[3]y[2] + x[2]y[3] \quad 4$$

$$z[-1] = x[0]y[6] + x[6]y[0] + x[3]y[3] + x[2]y[4] + x[4]y[2] + x[1]y[5] + x[5]y[1] \quad 4$$

$$z[0] = \alpha_0 \quad 8$$

$$z[1] = \alpha_1 - (x[0]y[6] + x[6]y[0] + x[2]y[4] + x[4]y[2]) + x[4]y[4] + x[2]y[6] + x[6]y[2] \quad 2$$

$$z[2] = \alpha_2 - (x[0]y[5] + x[5]y[0] + x[1]y[4] + x[4]y[1]) + x[4]y[5] + x[5]y[4] \quad 2$$

$$z[3] = \alpha_3 - (x[0]y[4] + x[4]y[0] + x[1]y[5] + x[5]y[1] + x[2]y[6] + x[6]y[2]) + \quad 4$$
$$+ x[5]y[5] + x[4]y[6] + x[6]y[4]$$

$$z[4] = \alpha_4 - (x[0]y[3] + x[3]y[0] + x[1]y[2] + x[2]y[1]) \quad 2$$

$$z[5] = \alpha_5 - (x[0]y[2] + x[2]y[0] + x[1]y[3] + x[3]y[1] + x[4]y[6] + x[6]y[4] + x[6]y[6]) \quad -$$

$$z[6] = \alpha_6 - (x[0]y[1] + x[1]y[0] + x[3]y[2] + x[2]y[3] + x[4]y[5] + x[5]y[4]) \quad -$$

$$z[7] = \alpha_7 - (x[0]y[0] + x[1]y[1] + x[2]y[2] + x[3]y[3] + x[4]y[4] + x[5]y[5] + x[6]y[6]) \quad -$$

Итого требуется 43 умножения и 127 сложений. Однако, если вернуться к результатам алгоритма частичной декомпозиции при линейной свертке 4×4 $(9M + 30A)$, то окажется, что задачу для последовательностей 8×8 выгоднее решать полной декомпозицией на 4 задачи для последовательностей 4×4. В этом случае, с учетом 13 операций сложения для стыковки, всего для линейной свертки последовательностей 8×8 необходимо 36 умножений и 140 сложений.

Полученные результаты говорят о том, что при синтезе алгоритмов линейной свертки для больших последовательностей требуется рассматривать различные варианты структуры алгоритма.

4.5 Циклическая свертка с использованием ГЧС

Циклическая свертка двух числовых последовательностей рассчитывается по формуле (4.8). Как и для линейной свертки, будем рассматривать числовые последовательности длиной 2^n. Из (4.8) следует, что для непосредственного расчета 2^n отсчетов циклической свертки необходимо выполнить 2^{2n} умножений и $2^n\left(2^n - 1\right)$ сложений. Представление числовых последовательностей длиной 2^n гиперкомплексными числами размерности длиной 2^n, принадлежащими ГЧС $W^{(n)}\left(e, 2^n\right)$, позволяет значительно снизить количество вещественных операций. Рассмотрим некоторые случаи.

1) $n=1$

Этот случай, по сути, был уже рассмотрен в разделе 4.3. Действительно, пусть необходимо вычислить циклическую свертку двух числовых последовательностей – $\{x[0],x[1]\}$ и $\{y[0],y[1]\}$. В соответствии с (4.8) свертка будет иметь два отсчета:

$$\begin{aligned} z[0] &= x[0]y[0]+x[1]y[1];\\ z[1] &= x[0]y[1]+x[1]y[0]. \end{aligned} \tag{4.89}$$

Как видно, для вычисления данной свертки необходимо выполнить 4 вещественных умножения и 2 сложения.

Будем, как и ранее, рассматривать элементы последовательностей $\{x[0],x[1]\}$ и $\{y[0],y[1]\}$ как компоненты гиперкомплексных чисел, принадлежащие ГЧС размерности 2 $W(e,2)$:

$$X = x[0]e_0 + x[1]e_1; \tag{4.90}$$

$$Y = y[0]e_0 + y[1]e_1. \tag{4.91}$$

В соответствии с таблицей умножения ГЧС W:

$$XY = (x[0]y[0]+x[1]y[1])e_0 + (x[0]y[1]+x[1]y[0])e_1. \tag{4.92}$$

Правая часть выражения (4.92) содержит все компоненты, необходимые для формирования свертки (4.89). Компонента при e_0 полностью совпадает с отсчетом $z[0]$, а компонента при e_1 – с отсчетом $z[1]$. Но количество вещественных умножений также равно 4, как и в (4.89). Таким образом, возникает необходимость уменьшить число умножений в (4.92).

Для уменьшения количества умножений целесообразно перейти из системы $W(e,2)$ в систему $W_1(f,2)$:

$$X_1 = (x[0]+x[1])f_0 + (x[0]-x[1])f_1; \tag{4.93}$$

$$Y_1 = (y[0]+y[0])f_0 + (y[0]-y[1])f_1. \tag{4.94}$$

Такой переход требует дополнительно 4 сложения. Однако, учитывая, что обычно числовая последовательность сворачивается с постоянным ядром, переход элементов которого в $W_1(f,2)$ сделан заранее, то можно считать, что переход из системы $W(e,2)$ в систему $W_1(f,2)$ требует только 2 сложения.

Умножение чисел (4.93) и (4.94) в соответствии с таблицей умножения $W_1(f,2)$:

$$X_1Y_1 = \alpha_0 f_0 + \alpha_1 f_1 = (x[0]+x[1])(y[0]+y[1])f_0 + (x[0]-x[1])(y[0]-y[1])f_1 \tag{4.95}$$

требует 2 умножения.

Обратный переход из $W_1(f,2)$ в систему $W(e,2)$ в соответствии с правым преобразованием (1.36) будет иметь вид:

$$XY = \frac{\alpha_0+\alpha_1}{2}e_0 + \frac{\alpha_0-\alpha_1}{2}e_1, \tag{4.96}$$

что также требует 2 сложения. Деление на 2 – это короткая операция, требующая только сдвига регистра. Поэтому ее можно не учитывать.

Приравнивая правые части (4.62) и (4.66), получим:

$$x[0]y[1]+x[1]y[0]=\frac{\alpha_0+\alpha_1}{2}, \tag{4.97}$$

$$x[0]y[0]+x[1]y[1]=\frac{\alpha_0-\alpha_1}{2}. \tag{4.98}$$

Таким образом, для вычисления отсчетов циклической свертки в случае $n=1$ требуется 2 умножения и 4 сложения.

2) $n=2$

Циклическая свертка числовых последовательностей длиной 4 $\{x[0],x[1],x[2],x[3]\}$ и $\{y[0],y[1],y[2],y[3]\}$ будет иметь четыре отсчета:

$$\begin{aligned}
z[0]&=x[0]y[0]+x[1]y[1]+x[2]y[2]+x[3]y[3];\\
z[1]&=x[0]y[1]+x[1]y[2]+x[2]y[3]+x[3]y[0];\\
z[2]&=x[0]y[2]+x[1]y[3]+x[2]y[0]+x[3]y[1];\\
z[3]&=x[0]y[3]+x[1]y[0]+x[2]y[1]+x[3]y[2].
\end{aligned} \tag{4.99}$$

Для вычислений (4.99) необходимо выполнить 16 умножений и 12 сложений.

Представим числовые последовательности $\{x[0],x[1],x[2],x[3]\}$ и $\{y[0],y[1],y[2],y[3]\}$ в виде гиперкомплексных чисел из системы $W^{(2)}(e,4)$:

$$X=\sum_{i=0}^{3}x[i]e_i; \tag{4.100}$$

$$Y=\sum_{i=0}^{3}y[i]e_i. \tag{4.101}$$

Тогда в соответствии с таблицей умножения ГЧС $W^{(2)}(e,4)$ произведение XY имеет вид

$$XY=\sum_{i=0}^{3}\alpha_i e_i\,, \tag{4.102}$$

где компоненты α_i таковы:

$$\begin{aligned}
\alpha_0&=x[0]y[0]+x[1]y[1]+x[2]y[2]+x[3]y[3];\\
\alpha_1&=x[0]y[1]+x[1]y[0]+x[2]y[3]+x[3]y[2];\\
\alpha_2&=x[0]y[2]+x[1]y[3]+x[2]y[0]+x[3]y[1];\\
\alpha_3&=x[0]y[3]+x[1]y[2]+x[2]y[1]+x[3]y[0].
\end{aligned}. \tag{4.103}$$

Для вычисления компонентов $\alpha_0,\alpha_1,\alpha_2,\alpha_3$ необходимо, как и в (4.99) выполнить 16 умножений и 12 сложений. Для уменьшения количества умножений можно, как описано в разделе 4.3, перевести числа X и Y из ГЧС $W^{(2)}(e,4)$ в изоморфную диагональную ГЧС $W_1^{(2)}(f,4)$, выполнить там умножение, и полученное произведение перевести обратно в ГЧС $W^{(2)}(e,4)$, получив тем самым число XY, компоненты которого и есть $\alpha_0,\alpha_1,\alpha_2,\alpha_3$. На это необходимо уже всего 4 умножения и 16 сложений.

Если обозначить

$$\beta = x[0]y[1] + x[2]y[3], \quad \gamma = x[1]y[2] + x[3]y[0],$$

то из сравнения (4.99) и (4.103) можно установить:

$$\begin{aligned} z[0] &= \alpha_0; \\ z[1] &= \beta + \gamma; \\ z[2] &= \alpha_2; \\ z[3] &= \alpha_1 + \alpha_3 - (\beta + \gamma). \end{aligned} \tag{4.104}$$

Для этих вычислений требуется 4 умножения и 3 сложения. Всего на все вычисления отсчетов циклической свертки четвертого порядка необходимо 8 умножений и 19 сложений.

3) $n = 3$

Для упрощения вида выкладок и экономии места будем использовать индексную форму записи суммы парных произведений, которая заключается в том, что вместо произведения $x[i]y[j]$ будем записываться только индексы сомножителей через запятую. Для обозначения индексной записи будем ее брать в круглые скобки, а вместо знака равенства ставить значки $\triangleright$ и $\triangleleft$, как показано ниже:

$$\beta = x[0]y[1] + x[2]y[3] \triangleright (0,1+2,3); \quad (1,2+3,0) \triangleleft x[1]y[2] + x[3]y[0].$$

При такой двухзначной индексации запись приобретает необходимое в данном случае свойство непозиционности, а значит, и возможность перестановки слагаемых.

Циклическая свертка числовых последовательностей длиной 8 $\{x[0],...,x[7]\}$ и $\{y[0],...,y[7]\}$ будет иметь 8 отсчетов и в индексной форме записи показана на рис. 4.6.

Для вычислений (4.105) необходимо выполнить 64 умножения и 56 сложений.

Представим числовые последовательности $\{x[0],...,x[7]\}$ и $\{y[0],...,y[7]\}$ в виде гиперкомплексных чисел из системы $W^{(3)}(e,8)$:

$$X = \sum_{i=0}^{7} x[i]e_i. \tag{4.105}$$

$$Y = \sum_{i=0}^{7} y[i]e_i. \tag{4.106}$$

$$
\begin{aligned}
z[0] &\triangleright (0,0+1,1+2,2+3,3+4,4+5,5+6,6+7,7);\\
z[1] &\triangleright (0,1+1,2+2,3+3,4+4,5+5,6+6,7+7,0);\\
z[2] &\triangleright (0,2+1,3+2,4+3,5+4,6+5,7+6,0+7,1);\\
z[3] &\triangleright (0,3+1,4+2,5+3,6+4,7+5,0+6,1+7,2);\\
z[4] &\triangleright (0,4+1,5+2,6+3,7+4,0+5,1+6,2+7,3);\\
z[5] &\triangleright (0,5+1,6+2,7+3,0+4,1+5,2+6,3+7,4);\\
z[6] &\triangleright (0,6+1,7+2,0+3,1+4,2+5,3+6,4+7,5);\\
z[7] &\triangleright (0,7+1,0+2,1+3,2+4,3+5,4+6,5+7,6).
\end{aligned}
\qquad (4.107)
$$

Рис. 4.6. Отсчеты циклической свертки 8-го порядка в индексной форме записи

Тогда в соответствии с таблицей умножения ГЧС $W^{(2)}(e,4)$ произведение XY имеет вид

$$
XY = \sum_{i=0}^{7} \alpha_i e_i \,, \qquad (4.108)
$$

где компоненты α_i таковы:

$$
\begin{aligned}
\alpha_0 &\triangleright (0,0+1,1+2,2+3,3+4,4+5,5+6,6+7,7);\\
\alpha_1 &\triangleright (0,1+1,0+2,3+3,2+4,5+5,4+6,7+7,6);\\
\alpha_2 &\triangleright (0,2+1,3+2,0+3,1+4,6+5,7+6,4+7,5);\\
\alpha_3 &\triangleright (0,3+1,2+2,1+3,0+4,7+5,6+6,5+7,4);\\
\alpha_4 &\triangleright (0,4+1,5+2,6+3,7+4,0+5,1+6,2+7,3);\\
\alpha_5 &\triangleright (0,5+1,4+2,7+3,6+4,1+5,0+6,3+7,2);\\
\alpha_6 &\triangleright (0,6+1,7+2,4+3,5+4,2+5,3+6,0+7,1);\\
\alpha_7 &\triangleright (0,7+1,6+2,5+3,4+4,3+5,2+6,1+7,0).
\end{aligned}
\qquad (4.109)
$$

Из сравнения (4.107) и (4.109) видно, что два отсчета свертки определяются непосредственно, а именно:

$$
z[0] = \alpha_0 ; \quad z[4] = \alpha_4 .
$$

Остальные отсчеты определяются из следующих соображений. Введем такие обозначения:

$$
\begin{aligned}
\beta_1 &\triangleright (1,0+3,2+5,4+7,6);\\
\beta_2 &\triangleright (1,2+5,6);\\
\beta_3 &\triangleright (3,4+7,0);\\
\beta_4 &\triangleright (2,0+3,1+6,4+7,5);\\
\beta_5 &\triangleright (2,4+3,5+6,0+7,1);\\
\beta_6 &\triangleright (1,4+3,6+5,0+7,2);\\
\beta_7 &\triangleright (0,3+4,7);
\end{aligned}
$$

$$\beta_8 \triangleright (2{,}5+6{,}1);$$
$$\beta_9 \triangleright (1{,}6+5{,}2);$$
$$\beta_{10} \triangleright (3{,}0+7{,}4).$$

Тогда

$$z[1] \triangleright \alpha_1 - \beta_1 + \beta_2 + \beta_3;$$
$$z[2] \triangleright \alpha_2 - \beta_4 + \beta_5;$$
$$z[3] \triangleright \beta_6 + \beta_7 + \beta_8;$$
$$z[5] \triangleright \alpha_5 - \beta_6 + \beta_9 + \beta_{10};$$
$$z[6] \triangleright \alpha_6 - \beta_5 - \beta_4;$$
$$z[7] \triangleright \alpha_3 + \alpha_7 + \beta_1 - \beta_2 - \beta_3 - \beta_7 - \beta_8 - \beta_9 - \beta_{10}.$$

Эти вычисления требуют 28 умножений и 34 сложения.

Для вычисления компонентов $\alpha_0,...,\alpha_7$ необходимо, как и в (4.107) выполнить 64 умножения и 56 сложений. Для уменьшения количества умножений можно, как описано в разделе 4.3, перевести числа X и Y из ГЧС $W^{(3)}(e,8)$ в изоморфную диагональную ГЧС $W_1^{(3)}(f,8)$, выполнить там умножение, и полученное произведение перевести обратно в ГЧС $W^{(3)}(e,8)$, получив тем самым число XY, компоненты которого и есть $\alpha_0,...,\alpha_7$. На это необходимо уже всего 8 умножений и 32 сложения.

Итак, всего для расчета восьми отсчетов циклической свертки требуется 36 умножений и 66 сложений.

4.6 Алгоритмы свертки, основанные на циклических сдвигах исходной числовой последовательности

В разделе 4.3 в конкретных примерах встречалась ситуация, когда два отсчета циклической свертки полностью совпадали с компонентами произведения гиперкомплексных чисел, являющимися представлениями сворачиваемых числовых последовательностей. В данном разделе будет показано, что эта ситуация будет встречаться при любой длине сворачиваемых последовательностей 2^n, а также исследованы пути увеличения количества таких совпадений и методы дальнейшего уменьшения количества операций, необходимых для вычисления циклической свертки.

4.6.1 Существование двух отсчетов циклической свертки порядка 2^n в гиперкомплексном произведении представлений сворачиваемых последовательностей

Рассмотрим процедуру удвоения Грассмана-Клиффорда, описанную в главе 1, в общем случае, то есть удвоение системы $W^{(n-1)}(g,2^{n-1})$ системой $W(e,2)$:

$$W^{(n)}\left(eg,2^n\right)=Д\left(W(e,2),W^{(n-1)}\left(g,2^{n-1}\right)\right).$$

Схема построения таблицы умножения системы $W^{(n)}\left(eg,2^n\right)$ имеет вид:

$W^{(n)}\left(eg,2^n\right)$	e_1g_1	e_1g_2	…	$e_1g_{2^{n-1}}$	e_2g_1	e_2g_2	…	$e_2g_{2^{n-1}}$
e_1g_1	$e_1e_1W^{(n-1)}\left(g,2^{n-1}\right)$				$e_1e_2W^{(n-1)}\left(g,2^{n-1}\right)$			
e_1g_2								
…								
$e_1g_{2^{n-1}}$								
e_2g_1	$e_1e_2W^{(n-1)}\left(g,2^{n-1}\right)$				$e_2e_2W^{(n-1)}\left(g,2^{n-1}\right)$			
e_2g_2								
…								
$e_2g_{2^{n-1}}$								

(4.110)

Перейдем в (4.110) к одно-символьному обозначению базиса и рассмотрим строение главной диагонали таблицы умножения системы $W^{(n)}\left(m,2^n\right)=W^{(n)}\left(eg,2^n\right)$, где базис $m=\{m_0,...,m_{2^n-1}\}$ есть результат перемножения базисов $e=\{e_1,e_2\}$ и $g=\{g_1,...,g_{2^{n-1}}\}$, а также двух ее боковых поддиагоналей, являющихся, в свою очередь, главными диагоналями систем $W^{(n-1)}\left(g,2^{n-1}\right)$. Такая таблица умножения показана ниже.

Из (4.113) видно, что если умножаются два числа $X=\sum_{i=0}^{2^n-1}x[i]m_i$ и $Y=\sum_{i=0}^{2^n-1}y[i]m_i$, то компоненты произведения $X\cdot Y$ при базисных элементах m_0 и $m_{2^{n-1}}$ будут соответственно такими:

$$\alpha_0=\sum_{i=0}^{2^n-1}x[i]\cdot y[i], \tag{4.111}$$

$$\alpha_{2^{n-1}}=\sum_{i=0}^{2^{n-1}-1}x[i]\cdot y\left[i+2^{n-1}\right]+\sum_{i=2^{n-1}}^{2^n-1}x[i]\cdot y\left[i-2^{n-1}\right]. \tag{4.112}$$

$W^{(n)}\left(m,2^n\right)$	m_0	m_1	…	$m_{2^{n-1}-1}$	$m_{2^{n-1}}$	$m_{2^{n-1}+1}$	…	m_{2^n-1}
m_0	m_0	m_1			$m_{2^{n-1}}$			
m_1	m_1	m_0				$m_{2^{n-1}}$		
…			…				…	
$m_{2^{n-1}-1}$				m_0				$m_{2^{n-1}}$
$m_{2^{n-1}}$	$m_{2^{n-1}}$				m_0			
$m_{2^{n-1}+1}$		$m_{2^{n-1}}$				m_0		
…			…				…	
m_{2^n-1}				$m_{2^{n-1}}$				m_0

(4.113)

Но такую же структуру имеют и отсчеты циклической свертки $z[0]$ и $z\left[2^{n-1}\right]$, то есть, вычислив компоненты α_0 и $\alpha_{2^{n-1}}$, мы тут же без дополнительных операций получаем два отсчета свертки. Можно показать, что других подобных случаев не может быть. Действительно, если упорядочить слагаемые отсчетов циклической свертки 2^n-го порядка по индексам сомножителя x, то индексы второго сомножителя y будут представлять собой циклические ряды упорядоченных индексов от 0 до 2^n-1:

$$\begin{aligned}
&z[0]: 0\ \ 1\ \ 2\ \ 3\ \ \ldots\ 2^n-2\ \ 2^n-1;\\
&z[1]\ : 1\ \ 2\ \ 3\ \ \ldots\ 2^n-2\ \ 2^n-1\ 0;\\
&z[2]: 2\ \ 3\ \ \ldots\ 2^n-2\ \ 2^n-1\ 0\ \ 1;\\
&z[3]: 3\ \ \ldots\ 2^n-2\ \ 2^n-1\ 0\ \ 1\ \ 2;\\
&z\left[2^n-1\right]: 2^n-1\ \ 0\ \ 1\ \ 2\ \ \ldots\ \ 2^n-2.
\end{aligned} \qquad (4.114)$$

В компонентах произведения $X\cdot Y$ такая цикличность как раз прослеживается только в 0-ом и 2^{n-1}-ом компонентах. Например, в 1-ом компоненте этот ряд индексов сразу начинается «неправильно»: 1 0 … .

4.6.2 Циклические сдвиги и точные отсчеты свертки

Как уже отмечено выше, для повышения быстродействия алгоритма свертки целесообразно увеличить количество компонентов произведения $X\cdot Y$, которые совпадали бы с другими отсчетами циклической свертки.

Рассмотрим циклически сдвинутую на ν позиций числовую последовательность $\{y[i]\}_{i=0}^{2^n-1}$ длиной 2^n. То есть, если исходная последовательность есть $\{y[i]\}_{i=0}^{2^n-1}$, то сдвинутая – $\{y\left[(i+\nu)\ \left(mod\ 2^n\right)\right]\}_{i=0}^{2^n-1}$.

Введем следующие обозначения:

– $Y^{(\nu)}\in W^{(n)}\left(e,2^n\right)$ гиперкомплексное число, соответствующее сдвинутой на ν отсчетов числовой последовательности;

– $Y_1^{(\nu)}\in W_1^{(n)}\left(f,2^n\right)$ – его образ в $W_1^{(n)}\left(f,2^n\right)$;

– $Y_1^{(\nu)}[k]$ – k-ый компонент образа.

Определим компоненты произведения $X \cdot Y^{(\nu)}$ в ГЧС $W^{(n)}\left(e,2^n\right)$ при базисных элементах e_0 и $e_{2^{n-1}}$:

$$\alpha_0^{(\nu)} = \sum_{i=0}^{2^n-1} x[i] \cdot y\left[(i+\nu)\ \left(mod\ 2^n\right)\right]. \qquad (4.115)$$

Если (4.115) сравнить с (4.114), то можно установить, что при каждом сдвиге ν определяются два отсчета свертки

$$z[\nu] = \alpha_0^{(\nu)}, \qquad (4.116)$$

$$z\left[\left(2^{n-1}+\nu\right)\ \left(mod\ 2^n\right)\right] = \alpha_{2^{n-1}}^{(\nu)}. \qquad (4.117)$$

Из (4.116) и (4.117) видно, что целесообразно делать не более 2^{n-1} сдвигов, так как результаты начинают повторяться.

Таким образом, с помощью таких сдвигов можно построить очень простой по структуре алгоритм вычисления циклической свертки, который заключается в следующем. Делается 2^{n-1} сдвигов (с учетом нулевого сдвига) одной из последовательностей. Для каждого сдвига вычисляется произведение сворачиваемых последовательностей путем перехода из $W^{(n)}\left(e,2^n\right)$ в $W_1^{(n)}\left(f,2^n\right)$ и обратно. На это тратится 2^n умножений и получаются два отсчета свертки. Итого необходимо 2^{2n-1} умножений, что в 2 раза меньше, чем при непосредственном вычислении свертки (2^{2n}).

4.6.3 Симметрии оператора изоморфизма систем $W^{(n)}\left(e,2^n\right)$ в $W_1^{(n)}\left(f,2^n\right)$ при циклическом сдвиге преобразуемого числа

Симметрии, существующие в операторе изоморфизма систем $W^{(n)}\left(e,2^n\right)$ в $W_1^{(n)}\left(f,2^n\right)$ позволяют уменьшить количество операций при вычислении циклической свертки числовых последовательностей.

Рассмотрим процесс построения оператора изоморфизма $L(\{e\},\{f\})$ базисов систем $W^{(n)}\left(e,2^n\right)$ и $W_1^{(n)}\left(f,2^n\right)$. Как видно из (1.48), оператор $L^{(n)}(\{e\},\{f\})$, который строится на n-ом этапе, представляет собой процесс рекуррентного построчного умножения оператора (1.46) на оператор, полученный на предыдущем этапе. Символически это можно представить так:

$$L^{(n)} = \begin{Bmatrix} (e_0 + e_1) \cdot L^{(n-1)} \\ (e_0 - e_1) \cdot L^{(n-1)} \end{Bmatrix}. \qquad (4.118)$$

Такое рекуррентное построение оператора преобразования базиса вносит определенные закономерности в его структуру в смысле чередования знаков в каждой строке. При этом следует учесть, что гиперкомплексные числа в системах $W^{(n)}\left(e,2^n\right)$ и $W_1^{(n)}\left(f,2^n\right)$ преобразуются по таким же законам. Будем при умножении в (4.118) придерживаться такого правила: сначала умножается строка

оператора на первое слагаемое, потом на второе и линейно нумеруются полученные элементы удвоенного базиса.

При таком удвоении оператора первые четыре компоненты 2^n-мерного оператора будут иметь вид:

$$\begin{aligned} e_0 &= f_0 + f_1 + f_2 + f_3 + ... + f_{2^n-4} + f_{2^n-3} + f_{2^n-2} + f_{2^n-1}; \\ e_1 &= f_0 - f_1 + f_2 - f_3 + ... + f_{2^n-4} - f_{2^n-3} + f_{2^n-2} - f_{2^n-1}; \\ e_1 &= f_0 + f_1 - f_2 - f_3 + ... + f_{2^n-4} + f_{2^n-3} - f_{2^n-2} - f_{2^n-1}; \\ e_1 &= f_0 - f_1 - f_2 + f_3 + ... + f_{2^n-4} + f_{2^n-3} - f_{2^n-2} - f_{2^n-1}. \end{aligned} \quad (4.119)$$

Из (4.119) можно записать, например, такие соотношения для сдвинутых числовых последовательностей:

$$Y_1^{(\nu)}[0] = Y_1^{(0)}[0], \quad (4.120)$$

где $Y_1^{(0)}[0]$ – компонент образа числа, соответствующий несдвинутой последовательности (нулевой сдвиг).

$$Y_1^{(\nu)}[1] = (-1)^{\nu} Y_1^{(0)}[0]. \quad (4.121)$$

$$Y_1^{(\nu=2k)}[2] = Y_1^{(0)}[2]. \quad (4.122)$$

$$Y_1^{(\nu=2k)}[3] = Y_1^{(0)}[2]. \quad (4.123)$$

Соотношений такого типа может быть и гораздо больше, но проследить этот процесс в общем виде достаточно тяжело. Гораздо проще это проделать для конкретных размерностей, что приведено ниже.

Ценность соотношений типа (4.120) – (4.123) заключается в том, что если $Y_1^{(\nu)}[i] = Y_1^{(\mu)}[i]$, $\iota > \mu$, то $X_1[i] \cdot Y_1^{(\nu)}[i] = X_1[i] \cdot Y_1^{(\mu)}[i]$, то есть произведение $X_1[i] \cdot Y_1^{(\nu)}[i]$ уже вычислено для одного из предыдущих сдвигов, то это умножение не требуется.

4.6.4 Алгоритм циклической свертки с использованием симметрии оператора изоморфизма в случае $2^n = 4$

Рассмотрим случай циклической свертки числовых последовательностей длиной $2^n = 4$. Будем, как и ранее, рассматривать элементы последовательностей $X = \{x[0], x[1], x[2], x[3]\}$ и $Y = \{y[0], y[1], y[2], y[3]\}$ как компоненты гиперкомплексных чисел ГЧС $W^{(2)}(e,4)[3]$:

$$X = x[0]e_0 + x[1]e_1 + x[2]e_2 + x[3]e_3, \quad (4.124)$$

$$Y = y[0]e_0 + y[1]e_1 + y[2]e_2 + y[3]e_3. \quad (4.125)$$

Тогда в соответствии с таблицей умножения ГЧС $W^{(2)}(e,4)$ произведение (4.110) и (4.111) имеет вид:

$$XY = \sum_{i=0}^{3} \alpha_i e_i, \quad (4.126)$$

а компоненты α_i:

$$\begin{aligned}
\alpha_0 &= x[0]y[0]+x[1]y[1]+x[2]y[2]+x[3]y[3];\\
\alpha_1 &= x[0]y[1]+x[1]y[0]+x[2]y[3]+x[3]y[2];\\
\alpha_2 &= x[0]y[2]+x[1]y[3]+x[2]y[0]+x[3]y[1];\\
\alpha_3 &= x[0]y[3]+x[1]y[2]+x[2]y[1]+x[3]y[0].
\end{aligned} \qquad (4.127)$$

Циклическая свертка этих числовых последовательностей будет иметь четыре отсчета (4.99), которые для удобства приведем еще раз:

$$\begin{aligned}
z[0] &= x[0]y[0]+x[1]y[1]+x[2]y[2]+x[3]y[3];\\
z[1] &= x[0]y[1]+x[1]y[2]+x[2]y[3]+x[3]y[0];\\
z[2] &= x[0]y[2]+x[1]y[3]+x[2]y[0]+x[3]y[1];\\
z[3] &= x[0]y[3]+x[1]y[2]+x[2]y[1]+x[3]y[0].
\end{aligned} \qquad (4.128)$$

Сравнивая (4.127) и (4.128), можно установить, что два отсчета свертки совпадают с компонентами произведения (4.112), а именно:

$$z[0]=\alpha_0; \quad z[2]=\alpha_2. \qquad (4.129)$$

Но для их вычисления требуется 4 умножения и 8 сложений, а, кроме того, еще необходимо вычислить оставшиеся два отсчета.

Перейдем из ГЧС $W^{(2)}(e,4)$ в изоморфную диагональную ГЧС $W_1^{(2)}(f,4)$. Для этого запишем преобразования (4.75) и (4.76) в матричной форме:

$$X_1=\begin{vmatrix} +1 & +1 & +1 & +1\\ +1 & -1 & +1 & -1\\ +1 & +1 & -1 & -1\\ +1 & -1 & -1 & +1 \end{vmatrix}\cdot\begin{vmatrix} x[0]\\ x[1]\\ x[2]\\ x[3]\end{vmatrix} \quad Y_1^{(0)}=\begin{vmatrix} +1 & +1 & +1 & +1\\ +1 & -1 & +1 & -1\\ +1 & +1 & -1 & -1\\ +1 & -1 & -1 & +1 \end{vmatrix}\cdot\begin{vmatrix} y[0]\\ y[1]\\ y[2]\\ y[3]\end{vmatrix}=\begin{vmatrix} Y_1^{(0)}[0]\\ Y_1^{(0)}[1]\\ Y_1^{(0)}[2]\\ Y_1^{(0)}[3]\end{vmatrix}, \qquad (4.130)$$

перемножим покомпонентно, как это следует из таблицы умножения ГЧС $W_1^{(2)}(f,4)$, и получим матрицу-столбец

$$A^{(0)}=\begin{vmatrix} A^{(0)}[0]\\ A^{(0)}[1]\\ A^{(0)}[2]\\ A^{(0)}[3]\end{vmatrix}=\begin{vmatrix} X_1[0]\cdot Y_1^{(0)}[0]\\ X_1[1]\cdot Y_1^{(0)}[1]\\ X_1[2]\cdot Y_1^{(0)}[2]\\ X_1[3]\cdot Y_1^{(0)}[3]\end{vmatrix}. \qquad (4.131)$$

Совершая обратный переход от $W_1^{(2)}(f,4)$ к $W^{(2)}(e,4)$ и не выписывая короткие операции, получим с учетом (4.129):

$$\begin{aligned}
z[0] &= \left(X_1[0]\cdot Y_1^{(0)}[0]+X_1[1]\cdot Y_1^{(0)}[1]\right)+\left(X_1[2]\cdot Y_1^{(0)}[2]+X_1[3]\cdot Y_1^{(0)}[3]\right);\\
z[2] &= \left(X_1[0]\cdot Y_1^{(0)}[0]+X_1[1]\cdot Y_1^{(0)}[1]\right)-\left(X_1[2]\cdot Y_1^{(0)}[2]+X_1[3]\cdot Y_1^{(0)}[3]\right).
\end{aligned} \qquad (4.132)$$

На эти вычисления требуется 4 умножения и 4 сложения.

Преобразуем числовую последовательность Y путем циклического сдвига в последовательность $Y^{(1)}=\{y[1],y[2],y[3],y[0]\}$. Тогда (4.113) примет вид:

$$\begin{aligned}
\alpha_0^{(1)} &= x[0]y[1]+x[1]y[2]+x[2]y[3]+x[3]y[0];\\
\alpha_1^{(1)} &= x[0]y[2]+x[1]y[1]+x[2]y[0]+x[3]y[3];\\
\alpha_2^{(1)} &= x[0]y[3]+x[1]y[0]+x[2]y[1]+x[3]y[2];\\
\alpha_3^{(1)} &= x[0]y[0]+x[1]y[3]+x[2]y[2]+x[3]y[1].
\end{aligned} \qquad (4.133)$$

Сравнивая (4.133) и (4.114), можно установить, что два отсчета свертки совпадают с компонентами произведения (4.133), а именно:

$$z[1]=\alpha_0^{(1)};\quad z[3]=\alpha_2^{(1)}. \qquad (4.134)$$

Таким образом, нужно найти компоненты $\alpha_0^{(1)}$, $\alpha_2^{(1)}$ произведения $XY^{(1)}$. Ищем его, как и в предыдущем случае.

$$Y_1^{(1)}=\begin{vmatrix}+1 & +1 & +1 & +1\\ +1 & -1 & +1 & -1\\ +1 & +1 & -1 & -1\\ +1 & -1 & -1 & +1\end{vmatrix}\cdot\begin{vmatrix}y[0]\\ y[1]\\ y[2]\\ y[3]\end{vmatrix}=\begin{vmatrix}Y_1^{(1)}[0]\\ Y_1^{(1)}[1]\\ Y_1^{(1)}[2]\\ Y_1^{(1)}[3]\end{vmatrix}. \qquad (4.135)$$

Но из симметричности оператора изоморфизма следует (сравн. с (4.120), (4.121)):

$$Y_1^{(1)}[0]=Y_1^{(0)}[0];\ Y_1^{(1)}[1]=-Y_1^{(0)}[1];\ Y_1^{(1)}[2]=-Y_1^{(0)}[3];\ Y_1^{(1)}[3]=Y_1^{(0)}[2]. \qquad (4.136)$$

Равенство (4.131) примет вид:

$$A^{(1)}=\begin{vmatrix}A^{(1)}[0]\\ A^{(1)}[1]\\ A^{(1)}[2]\\ A^{(1)}[3]\end{vmatrix}=\begin{vmatrix}X_1[0]\cdot Y_1^{(0)}[0]\\ -X_1[1]\cdot Y_1^{(0)}[1]\\ -X_1[3]\cdot Y_1^{(0)}[3]\\ X_1[2]\cdot Y_1^{(0)}[2]\end{vmatrix}. \qquad (4.137)$$

Совершая обратный переход от системы $W_1^{(2)}(f,4)$ к $W^{(2)}(e,4)$, с учетом (4.134) получим:

$$\begin{aligned}
z[1] &= \left(X_1[0]\cdot Y_1^{(0)}[0]-X_1[1]\cdot Y_1^{(0)}[1]\right)-\left(X_1[2]\cdot Y_1^{(0)}[3]-X_1[3]\cdot Y_1^{(0)}[2]\right);\\
z[3] &= \left(X_1[0]\cdot Y_1^{(0)}[0]-X_1[1]\cdot Y_1^{(0)}[1]\right)+\left(X_1[2]\cdot Y_1^{(0)}[3]-X_1[3]\cdot Y_1^{(0)}[2]\right).
\end{aligned} \qquad (4.138)$$

Здесь первые два слагаемых в каждой формуле вычислены на этапе (4.132). Поэтому требуется всего 2 умножения и 4 сложения. Всего для определения всех отсчетов свертки с учетом переходов из $W_1^{(2)}(f,4)$ в $W^{(2)}(e,4)$ и обратно требуется 6 умножений и 24 сложения.

4.6.5 Алгоритм циклической свертки с использованием симметрии оператора изоморфизма в случае $2^n = 8$

Здесь также, как и в разделе 4.5 будем использовать индексную форму записи суммы парных произведений, которая заключается в том, что вместо

произведения $x[i]y[j]$ будем записываться только индексы сомножителей через запятую.

Циклическая свертка числовых последовательностей длиной 8 $\{x[0],...,x[7]\}$ и $\{y[0],...,y[7]\}$ будет иметь 8 отсчетов и в индексной форме имеет вид (4.107).

Как отмечалось, для вычислений (4.107) необходимо выполнить 64 умножения и 56 сложений.

Представим числовые последовательности $\{x[0],...,x[7]\}$ и $\{y[0],...,y[7]\}$ в виде гиперкомплексных чисел из системы $W^{(3)}(e,8)$

$$X = \sum_{i=0}^{7} x[i]e_i; \tag{4.139}$$

$$Y = \sum_{i=0}^{7} y[i]e_i. \tag{4.140}$$

Тогда в соответствии с таблицей умножения ГЧС $W^{(2)}(e,4)$ произведение XY имеет вид

$$XY = \sum_{i=0}^{7} \alpha_i e_i, \tag{4.141}$$

где компоненты α_i приведены в (4.109).

Из сравнения (4.107) и (4.109) видно, что два отсчета свертки определяются непосредственно, а именно:

$$z[0] = \alpha_0; \qquad z[4] = \alpha_4. \tag{4.142}$$

Но для их вычисления требуется 8 умножений 56 сложений, а, кроме того, еще необходимо вычислить оставшиеся отсчеты.

Перейдем из ГЧС $W^{(3)}(e,8)$ в изоморфную диагональную ГЧС $W_1^{(3)}(f,8)$. Для этого запишем преобразования (4.81) в матричной форме. Оператор изоморфизма имеет вид

$$L = \begin{vmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\ +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 \end{vmatrix}, \tag{4.143}$$

а числа преобразуются так:

$$X_1 = L \cdot (x[0],...,x[7])^T,$$

$$Y_1^{(0)} = L \cdot (y[0],...,y[7])^T = \left(Y_1^{(0)}[0],...,Y_1^{(0)}[7]\right)^T \tag{4.144}$$

перемножим покомпонентно, как это следует из таблицы умножения ГЧС $W_1^{(3)}(f,8)$, и получим матрицу-столбец.

$$A^{(0)}=\left(A^{(0)}[0],A^{(0)}[1],...,A^{(0)}[7]\right)^T=\left(X_1[0]\cdot Y_1^{(0)}[0]+X_1[1]\cdot Y_1^{(0)}[1]+...+X_1[7]\cdot Y_1^{(0)}[7]\right)^T. \quad (4.145)$$

Совершая обратный переход от $W_1^{(3)}(f,8)$ к $W^{(3)}(e,8)$ и не выписывая короткие операции, получим с учетом (4.142):

$$\begin{aligned}z[0]&=\left(X_1[0]\cdot Y_1^{(0)}[0]+X_1[1]\cdot Y_1^{(0)}[1]+X_1[2]\cdot Y_1^{(0)}[2]+X_1[3]\cdot Y_1^{(0)}[3]\right)+\\&+\left(X_1[4]\cdot Y_1^{(0)}[4]+X_1[5]\cdot Y_1^{(0)}[5]+X_1[6]\cdot Y_1^{(0)}[6]+X_1[7]\cdot Y_1^{(0)}[7]\right);\\z[0]&=\left(X_1[0]\cdot Y_1^{(0)}[0]+X_1[1]\cdot Y_1^{(0)}[1]+X_1[2]\cdot Y_1^{(0)}[2]+X_1[3]\cdot Y_1^{(0)}[3]\right)-\\&-\left(X_1[4]\cdot Y_1^{(0)}[4]+X_1[5]\cdot Y_1^{(0)}[5]+X_1[6]\cdot Y_1^{(0)}[6]+X_1[7]\cdot Y_1^{(0)}[7]\right).\end{aligned}$$

На эти вычисления требуется 8 умножений и 8 сложений.

Преобразуем числовую последовательность Y путем циклического сдвига в последовательность $Y^{(1)}=\{y[1],y[2],y[3],y[0]\}$. Тогда (4.109) примет вид:

$$\begin{aligned}&\alpha_0^{(1)}\triangleright(0,1+1,2+2,3+3,4+4,5+5,6+6,7+7,0);\\&\alpha_1^{(1)}\triangleright(0,2+1,1+2,4+3,3+4,6+5,5+6,0+7,7);\\&\alpha_2^{(1)}\triangleright(0,3+1,4+2,1+3,2+4,7+5,0+6,5+7,6);\\&\alpha_3^{(1)}\triangleright(0,4+1,3+2,2+3,1+4,0+5,7+6,6+7,5);\\&\alpha_4^{(1)}\triangleright(0,5+1,6+2,7+3,0+4,1+5,2+6,3+7,4);\\&\alpha_5^{(1)}\triangleright(0,6+1,5+2,0+3,7+4,2+5,1+6,4+7,3);\\&\alpha_6^{(1)}\triangleright(0,7+1,0+2,5+3,6+4,3+5,4+6,1+7,2);\\&\alpha_7^{(1)}\triangleright(0,0+1,7+2,6+3,5+4,2+5,3+6,2+7,1).\end{aligned} \quad (4.146)$$

Сравнивая (4.146) и (4.107), можно установить, что два отсчета свертки совпадают с компонентами произведения (4.133), а именно:

$$z[1]=\alpha_0^{(1)};\quad z[5]=\alpha_4^{(1)}. \quad (4.147)$$

Перейдем из ГЧС $W^{(3)}(e,8)$ в изоморфную диагональную ГЧС $W_1^{(3)}(f,8)$:

$$Y_1^{(1)}=L\cdot(y[1],y[2],...,y[7],y[0])^T=\left(Y_1^{(1)}[0],...,Y_1^{(1)}[7]\right)^T. \quad (4.148)$$

Но из структуры оператора L вытекают следующие равенства:

$$Y_1^{(1)}[0]=Y_1^{(0)}[0];\ Y_1^{(1)}[1]=-Y_1^{(0)}[1];\ Y_1^{(1)}[2]=-Y_1^{(0)}[3];\ Y_1^{(1)}[3]=Y_1^{(0)}[2]. \quad (4.149)$$

Перемножая покомпонентно с X_1, в соответствии с таблицей умножения ГЧС $W_1^{(3)}(f,8)$, и совершая обратный переход от $W_1^{(3)}(f,8)$ к $W^{(3)}(e,8)$, получим с учетом (4.142):

$$z[1]=\left(X_1[0]\cdot Y_1^{(0)}[0]-X_1[1]\cdot Y_1^{(0)}[1]-X_1[2]\cdot Y_1^{(0)}[3]+X_1[3]\cdot Y_1^{(0)}[2]\right)+\sum_{i=4}^{7}X_1[i]\cdot Y_1^{(1)}[i];$$

$$z[4]=\left(X_1[0]\cdot Y_1^{(0)}[0]-X_1[1]\cdot Y_1^{(0)}[1]-X_1[2]\cdot Y_1^{(0)}[3]+X_1[3]\cdot Y_1^{(0)}[2]\right)-\sum_{i=4}^{7}X_1[i]\cdot Y_1^{(1)}[i].$$

На эти вычисления требуется уже на два умножения меньше – 6 умножений, так как первые два слагаемых в обоих отсчетах уже рассчитаны при нулевом сдвиге. В третьем и четвертом слагаемом умножение необходимо, но то, что значения вторых сомножителей рассчитаны тоже на предыдущем этапе, сокращает число сложений.

Выполняя еще два сдвига и действуя таким же образом, как и при нулевом и первом сдвигах получим:

$$z[2]=\alpha_0^{(2)};\quad z[6]=\alpha_4^{(2)};$$
$$z[3]=\alpha_0^{(3)};\quad z[7]=\alpha_4^{(3)};$$
$$Y_1^{(2)}[0]=Y_1^{(0)}[0];\ Y_1^{(2)}[1]=Y_1^{(0)}[1];\ Y_1^{(2)}[2]=-Y_1^{(0)}[3];\ Y_1^{(2)}[3]=-Y_1^{(0)}[2];$$
$$Y_1^{(2)}[4]=-Y_1^{(0)}[7];\quad Y_1^{(2)}[6]=Y_1^{(0)}[4];\quad Y_1^{(2)}[7]=Y_1^{(0)}[5];$$
$$Y_1^{(3)}[0]=Y_1^{(0)}[0];\ Y_1^{(3)}[1]=-Y_1^{(0)}[1];\ Y_1^{(3)}[2]=Y_1^{(0)}[3];\ Y_1^{(3)}[3]=-Y_1^{(0)}[2];$$
$$Y_1^{(3)}[4]=-Y_1^{(1)}[6];\quad Y_1^{(3)}[6]=-Y_1^{(1)}[4];$$

И остальные отсчеты свертки будут:

$$z[2]=\left(X_1[0]\cdot Y_1^{(0)}[0]+X_1[1]\cdot Y_1^{(0)}[1]-X_1[2]\cdot Y_1^{(0)}[3]-X_1[3]\cdot Y_1^{(0)}[2]\right)-$$
$$-\left(X_1[4]\cdot Y_1^{(0)}[7]-X_1[5]\cdot Y_1^{(1)}[5]-X_1[6]\cdot Y_1^{(0)}[4]-X_1[7]\cdot Y_1^{(0)}[5]\right),$$
$$z[6]=\left(X_1[0]\cdot Y_1^{(0)}[0]+X_1[1]\cdot Y_1^{(0)}[1]-X_1[2]\cdot Y_1^{(0)}[3]-X_1[3]\cdot Y_1^{(0)}(2)\right)+$$
$$+\left(X_1[4]\cdot Y_1^{(0)}[7]-X_1[5]\cdot Y_1^{(1)}[5]-X_1[6]\cdot Y_1^{(0)}[4]-X_1[7]\cdot Y_1^{(0)}[5]\right).$$

На эти вычисления требуется еще меньше умножений – 4 умножения.

$$z[3]=\left(X_1[0]\cdot Y_1^{(0)}[0]-X_1[1]\cdot Y_1^{(0)}[1]+X_1[2]\cdot Y_1^{(0)}[3]-X_1[3]\cdot Y_1^{(0)}[2]\right)-$$
$$-\left(X_1[4]\cdot Y_1^{(1)}[6]-X_1[5]\cdot Y_1^{(3)}[5]+X_1[6]\cdot Y_1^{(1)}[4]-X_1[7]\cdot Y_1^{(3)}[7]\right);$$
$$z[7]=\left(X_1[0]\cdot Y_1^{(0)}[0]-X_1[1]\cdot Y_1^{(0)}[1]+X_1[2]\cdot Y_1^{(0)}[3]-X_1[3]\cdot Y_1^{(0)}[2]\right)+$$
$$+\left(X_1[4]\cdot Y_1^{(1)}[6]-X_1[5]\cdot Y_1^{(3)}[5]+X_1[6]\cdot Y_1^{(1)}[4]-X_1[7]\cdot Y_1^{(3)}[7]\right).$$

На эти вычисления также требуется 4 умножения.

Таким образом, для вычисления отсчетов циклической свертки последовательностей длиной 8 членов с использованием симметрии оператора изоморфизма требуется 22 умножения.

Такие алгоритмы могут быть синтезированы и для больших длин сворачиваемых последовательностей. Они отличаются простотой структуры и легко формализуются.

ГЛАВА 5 ПРИМЕРЫ МОДЕЛЕЙ НА БАЗЕ ГЧС

5.1 Модель информационного поиска с использованием теории гиперкомплексных числовых систем

В настоящее время существует несколько традиционных моделей поиска: булева, вероятностная, векторно-пространственная [50]. Ниже предлагается еще одна модель, которая базируется на применении гиперкомплексных числовых систем (ГЧС) и является развитием канонической векторно-пространственной модели. Наведенный подход позволяет учитывать логическую операцию отрицания, явления синонимии, омонимии и т.п.

Перейдем к формальному изложению. Пусть словарь термов (слов и устойчивых словосочетаний) содержит N термов: $T=\{t_1,t_2,...,t_N\}$. В соответствии с классической векторно-пространственной модели информационного поиска документ представляется в виде вектора весовых значений термов размерности: $N:\ \vec{d}=(w_1,w_2,...,w_N)$. При этом весовые значения могут быть как числами 0 или 1 (терм входит или не входит в документ), так и весовыми значениями в интервале [0,1], которые рассчитываются, например, по формуле TF IDF.

В модели информационного поиска с использованием ГЧС предполагается каждому терму приписывать весовые значения w_i^+ и w_i^-, которые соответствуют вхождению терма (w_i^+) или его невхождению (w_i^-). Оба значения могут быть в интервале [0,1], в отдельных случаях можно предположить, что $w_i^+ + w_i^- = 1$.

Предлагается использовать ГЧС размерности $2N$ с базисом $\{e,e,...,e_{2N}\}$ и законом умножения в виде таблицы:

	e_1	e_2	e_3	e_4	…	e_{2N-1}	e_{2N}
e_1	e_1	e_2	0	0	…	0	0
e_2	e_2	e_1	0	0	…	0	0
e_3	0	0	e_3	e_4	…	0	0
e_4	0	0	e_4	e_3	…	0	0
…	…	…	…	…	…	…	…
e_{2N-1}	0	0	0	0	0	e_{2N-1}	e_{2N}
e_{2N}	0	0	0	0	0	e_{2N}	e_{2N-1}

Модель документа рассматривается как гиперкомплексное число вида:

$$D=e_1w_1^+ + e_2w_1^- + e_3w_2^+ + e_4w_2^- + ... + e_{2N-1}w_N^+ + e_{2N}w_N^-.$$

Можно рассмотреть оценку близости двух документов A и B ($A=e_1a_1^+ + e_2a_1^- + ... + e_{2N-1}a_N^+ + e_{2N}a_N^-$, $B=e_1b_1^+ + e_2b_1^- + ... + e_{2N-1}b_N^+ + e_{2N}b_N^-$):

$$Est(A,B)=Norm\left(\frac{1}{N}\left(\sum_{i=1}^{N}\left(e_{2i-1}a_i^+ + e_{2i}a_i^-\right)\left(e_{2i-1}b_i^+ + e_{2i}b_i^-\right)\right)\right),$$

где $Norm(\ )$ – функция нормы гиперкомплексного числа: $Norm(e_{2i-1}) = e_1$, $Norm(e_{2i}) = -e_1$.

Запрос $Q = q_1 e_1 + q_2 e_2 + q_3 e_3 + ... + q_{2N} e_{2N}$ также представляется в виде гиперкомплексного числа, как и документ D. Чем больше значение $Est(Q, D)$, тем документ более релевантный запросу.

Можно ввести и другую оценку близости между гиперкомплексными числами, по аналогии с нормой различий между обычными векторами в векторном пространстве:

$$Est1(A, B) = Norm\left(\frac{1}{N} \left(\sum_{i=1}^{N} \left(e_{2i-1} a_i^+ - e_{2i-1} b_i^+ \right)^2 \left(e_{2i} a_i^- - e_{2i} b_i^- \right)^2 \right) \right).$$

В этом случае документ более релевантный запросу, если оценка $Est1(Q, D)$ меньше.

Вместе с этим, вторая оценка по содержанию менее пригодна для задач информационного поиска. Например, при сравнении документа с запросом, даже при полностью нулевых значениях весовых коэффициентов в запросе, сумма в приведенном для выражения $Est1(Q, D)$ не будет нулевой, т.е. полностью зависит от документа.

Поэтому можно ограничиться применением первой оценки [51].

5.2 Гиперкомплексный подход к моделированию репутации в социальных сетях

В последнее время в рамках теории анализа социальных сетей большое внимание уделяется оценке репутации отдельных субъектов (агентов, узлов социальных сетей) и уровня доверия между ними [52-54].

Формально социальная сеть представляет собой граф, в котором множество вершин – это совокупность агентов, субъектов – индивидуальных или коллективных, а множество ребер представляет собой совокупность отношений совокупности социальных связей между агентами.

При моделировании социальных сетей возникает необходимость учета динамики социальных связей – взаимного влияния агентов.

Влияние в данном случае рассматривается как процесс и результат изменения индивидом (субъектом влияния) поведения другого субъекта – объекта влияния, его установок и оценок в ходе взаимодействия [55]. Таким образом, влияние – это способность воздействовать на чьи-либо представления или действия [56]. Различают направленное и ненаправленное влияние [55]. Направленное влияние используют в качестве механизмов воздействия на другого человека, например, путем убеждения или внушения. При этом индивид – субъект влияния – ставит перед собой задачу добиться определенных результатов от объекта влияния. Ненаправленное (нецеленаправленное, косвенное) влияние – это влияние, при котором индивид не ставит перед собой задачу добиться определенных результатов от объекта влияния.

Целенаправленное влияние участников социальной сети (или субъектов, не входящих в сеть, но использующих ее в качестве инструмента информационного воздействия) является частным случаем информационного управления, заключающегося в формировании управляемых субъектов такой информированности [57], чтобы принимаемые ими на ее основе решения были наиболее выгодны для управляющего субъекта [58].

Возможности влияния одних участников социальной сети на других ее участников существенно зависят от репутации первых. Репутация – «создавшееся общее мнение о достоинствах или недостатках кого-либо, чего-либо, общественная оценка» [57]. Репутацию можно рассматривать как «весомость» мнения сообщества об отдельном агенте или группе агентов, определяемую его взглядами и деятельностью (активностью). При этом репутация может быть как индивидуальной, так и коллективной.

Репутация возрастает, если выбор агента (ответы на некоторые ключевые вопросы) совпадает с тем, чего от него ожидает сообщество, и понижается в противном случае.

Пусть $\{a_1, a_2, ..., a_n\}$ – множество агентов – узлов социальной сети, которые влияют друг на друга. Матрицу влияния обозначим как $A = \left\| a_{ij} \right\|_{i=1,n}^{j=1,n}$ ($a_{ij} \geq 0$ обозначает степень доверия *i*-го агента *j*-му). При этом очевидно, что если *i*-й агент влияет на *j*-го, а *j*-й влияет на *k*-го, то это означает следующее: *i*-й агент косвенно влияет на *k*-го (транзитивность), что позволяет строить цепочки косвенных влияний.

Предположим, что у каждого агента в начальный момент времени имеется мнение по некоторому ключевому вопросу. Пусть мнение сообщества агентов сети отражает вектор начальных мнений b^0 размерности *n*. Мнение каждого агента меняется под влиянием мнений других агентов социальной сети.

Будем считать, что мнение *i*-го агента в момент времени t равно

$$b_i^t = \sum_{j=1}^{n} a_{ji} b_j^{t-1} .$$

При многократном обмене мнениями, мнения агентов сходятся к результирующему вектору мнений $B = \lim_{t \to \infty} b^t$. [53] Таким образом, справедливо соотношение $B = Ab$.

Обозначим r_i – параметр, описывающий репутацию *i*-го агента в социальной сети (сообществе), которую можно определить как нормированную сумму его влияний на всех остальных агентов социальной сети (предполагается $a_{ij} \geq 0 \;\; i, j = 1,...,n$), т.е.:

$$r_i = \frac{\sum_{i \neq j} a_{ij}}{R}, \quad j = 1,...,n .$$

Здесь $R = \sum_k \sum_{j \neq k} a_{kj} \quad k, j = 1,...,n$ – суммарное взаимное влияние друг на друга всех членов социальной сети.

В соответствии с приведенным выражением агент i имеет тем более высокую репутацию, чем выше его влияние на всех остальных членов социальной сети.

Аналогично, как построена модель поиска, так же может быть построена и модель определения репутации с использованием ГЧС.

Значение коэффициентов при базисных элементах образов субъектов социальной сети могут соответствовать вероятностям позитивного (или негативного) отношения субъектов к соответствующим вопросам. В этом случае путем перенумерации базисных элементов таблицу умножения ГЧС можно представить в следующем виде:

<table>
<tr><td></td><td>e_1</td><td>e_2</td><td>e_3</td><td>e_4</td><td>…</td><td>e_{4N-3}</td><td>e_{4N-2}</td><td>e_{4N-1}</td><td>e_{4N}</td></tr>
<tr><td>e_1</td><td colspan="4" rowspan="4">B_1</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>e_2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>e_3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>e_4</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>e_{4N-3}</td><td></td><td></td><td></td><td></td><td></td><td colspan="4" rowspan="4">B_N</td></tr>
<tr><td>e_{4N-2}</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>e_{4N-1}</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>e_{4N}</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

где блок B_1 (а в общем случае и любой ненулевой блок) будет иметь вид, дополненный коэффициентами w_i^j, вычисляемыми в процессе обучения модели:

	e_1	e_2	e_3	e_4
e_1	$w_1^1 e_1$	$w_1^2 e_2$	$w_2^1 e_1$	$w_2^2 e_1$
e_2	$w_1^2 e_2$	$w_1^1 e_1$	$w_2^2 e_1$	$w_2^1 e_1$
e_3	$w_2^1 e_1$	$w_2^2 e_1$	$w_3^1 e_3$	$w_3^2 e_4$
e_4	$w_2^2 e_1$	$w_2^1 e_1$	$w_3^2 e_4$	$w_3^1 e_3$

При этом w_1^1 – вес позитивного отношения к вопросу t_1, w_1^2 – вес негативного отношения к вопросу t_1, w_1^2, w_2^2 – веса взаимосвязей наличия противоречивых одновременно позитивных и негативных отношений к вопросу t_1.

Тогда для агента $A = a_1 e_1 + a_2 e_2 + a_3 e_3 + a_4 e_4$ и агента $A = b_1 e_1 + b_2 e_2 + b_3 e_3 + b_4 e_4$ оценка близости будет иметь следующий вид:

$$Sim(A \cdot B) = Norm\left(\frac{1}{2}\left(\begin{array}{l} e_1\left(\begin{array}{l} w_1^1(a_1b_1 + a_2b_2) + w_2^1(a_1b_3 + a_2b_4 + a_3b_1 + a_4b_2) \\ + w_2^2(a_1b_4 + a_2b_3 + a_3b_2 + a_4b_1) \end{array}\right) + \\ + e_2\left(w_1^2(a_1b_2 + a_2b_1)\right) + e_3\left(w_3^1(a_3b_3 + a_4b_3)\right) + e_4\left(w_3^2(a_3b_4 + a_4b_4)\right) \end{array}\right)\right)$$

При этом функция *Norm* соответствует той ГЧС, таблица которой используется для поиска. Следует учитывать, что предложенная таблица может быть составной частью таблицы большей размерности. Переход от заполненной таблицы умножения к слабо-заполненной (разреженной) можно осуществить изоморфным переходом [1–2], что значительно сократит количество операций при вычислении функции близости между субъектами.

В общем случае уровень доверия (близость) между субъектами $A = e_1 a_1^+ + e_2 a_1^- + ... + e_{2N-1} a_N^+ + e_{2N} a_N^-$ и $B = e_1 b_1^+ + e_2 b_1^- + ... + e_{2N-1} b_N^+ + e_{2N} b_N^-$, который можно трактовать как степень доверия, также характеризуется функцией, приведенной выше с учетом возможного наличия ненулевых недиагональных элементов таблицы:

$$Sim(A,B) = Norm\left(\begin{array}{l}\frac{1}{N}\left(e_1 a_1^+ + e_2 a_1^- + ... + e_{2N-1} a_N^+ + e_{2N} a_N^-\right) + \\ + \left(e_1 b_1^+ + e_2 b_1^- + ... + e_{2N-1} b_N^+ + e_{2N} b_N^-\right)\end{array}\right).$$

Репутация субъекта $A = e_1 a_{i1}^+ + e_2 a_{i1}^- + ... + e_{2N-1} a_{iN}^+ + e_{2N} a_{iN}^-$ в рамках всей социальной сети (т.е. по отношению к обществу) при этом определяется, как нормированная сумма уровней доверия со всеми остальными субъектами:

$$Trust(A_i) = \frac{\sum_{j \neq i} Sim(A_i, A_j)}{\sum_{k \neq i} Sim(A_k, A_j)}.$$

Для оценки уровня взаимного влияния субъектов социальной сети могут также использоваться другие различные методы, среди которых можно выделить: расчет меры взаимной информации (mutual information), расчет модифицированного коэффициента Dice (modified Dice coefficient), вхождение правдоподобия (log likelihood ratio), оценку χ^2 (Chi-square test) [59–62]. Вместе с тем, без специальных модификаций никакой из них не позволяет учитывать одновременно уровень позитивного и негативного отношения одного субъекта к ключевым вопросам, взаимную зависимость ключевых вопросов, вплоть до смысловой синонимии.

5.3 Исследование моделей

Рассмотрим, для примера, некоторые упрощенные частные случаи, при которых анализируются отношения общества и субъекта к одному вопросу[63]:

1) Пусть значение, соответствующее обществу, имеет вид: $Q = \frac{1}{2} e_1 + \frac{1}{2} e_2$, т.е. отношение к выбранному вопросу в обществе может быть как позитивным, так и негативным, с равной вероятностью. Пусть отношение субъекта к этому же вопросу однозначно позитив-

ное, а именно: $D = e_1$. В этом случае $Est(Q, D) = \frac{1}{2} - \frac{1}{2} = 0$, что соответствует полной неопределенности.

2) Пусть значение, соответствующее обществу, как и в предыдущем случае имеет вид: $Q = \frac{1}{2}e_1 + \frac{1}{2}e_2$. Пусть отношение субъекта к этому же вопросу также имеет вид: $D = \frac{1}{2}e_1 + \frac{1}{2}e_2$. В этом случае $Est(Q, D) = \frac{1}{4} + \frac{1}{4} - \frac{1}{2} = 0$, что, как и в предыдущем случае, соответствует полной неопределенности.

3) Пусть значение, соответствующее обществу, имеет вид: $Q = e_1$, а отношение субъекта к этому же вопросу: $D = \frac{4}{5}e_1 + \frac{1}{5}e_2$, тогда $Est(Q, D) = \frac{4}{5} - \frac{1}{5} = \frac{3}{5}$.

Следует отметить, что не все нулевые элементы приведенной выше «идеальной» таблицы в реальности могут быть нулевыми, однако предполагается, что данная таблица будет разреженной. Редкие ненулевые элементы в ней могут характеризовать взаимосвязь различных вопросов.

Применение моделей информационного поиска и определения репутации в социальных сетях на базе использования ГЧС может обеспечивать: возможность обучения соответствующей системы, учет некоторой смысловой синонимии, омонимии в ключевых вопросах (под омонимией здесь понимается различные по смыслу ключевые вопросы, заданные примерно одним набором слов) на уровне расширения таблиц умножения соответствующих ГЧС; возможность применения имеющихся наработок в области гиперкомплексных числовых систем, в том числе изоморфных ГЧС, более пригодных для быстрых вычислений.

ПРИЛОЖЕНИЯ

П1 Структура и функции алгоритмически-программной системы вычислений в ГЧС в среде MAPLE

Исследование и реализация быстрых алгоритмов обработки информации с использованием гиперкомплексных чисел требует наличия мощной среды вычислений, которая может позволить проводить аналитические преобразования различной сложности.

MAPLE – это пакет для аналитический вычислений, содержащий несколько тысяч команд, которые позволяют решать задачи алгебры, геометрии, математического анализа, статистики, математической физики. Основными объектами в MAPLE являются формулы, состоящие из математических символов, и выполняемые над ними действия. То есть пользователи могут вводить выражения в традиционной математической, то есть символьной форме. Вследствие этого MAPLE называют также пакетом символьной математики.

Ядро системы MAPLE написано на языке C и реализует язык MAPLE. Основная часть функциональности системы реализована в десятках различных библиотек, большинство из которых написано на самом языке MAPLE. Символьные выражения хранятся в памяти в виде ориентированного ациклического графа. Стандартный интерфейс реализован на языке Java, и при этом существуют возможности его расширения.

В ходе работы над книгой создана библиотека процедур для системы MAPLE, содержащая процедуры, которые потребуются для работы с быстрыми алгоритмами обработки сигналов. Использование символьных выражений в MAPLE позволяет создавать более компактные и эффективные алгоритмы для работы с гиперкомплексными числами.

Библиотека использует гиперкомплексное представление чисел и представление таблицы умножения гиперкомплексной числовой системы в общем виде. Так, основные процедуры библиотеки могут быть использованы как для канонических, так и для неканонических ГЧС. Также, вычисления могут проводиться как в символьном, так и в численном виде, в зависимости от заданных коэффициентов.

Библиотеку можно условно разделить на две части – реализация основных операций и вычисление основных алгебраических свойств, и реализация непосредственно быстрых алгоритмов обработки информации.

Из операций в библиотеке реализованы: сложение, вычитание, умножение и деление. Часть библиотеки, включающая реализацию алгебраических свойств, содержит процедуры по вычислению единичного элемента гиперкомплексной числовой системы, нормы гиперкомплексного числа и произведения сопряженных гиперкомплексному числу, а так же проверку, является ли гиперкомплексное число делителем нуля. Также приведена процедура изоморфного

перевода числа из одной гиперкомплексной системы в другую заданной размерности.

Часть, включающая непосредственно быстрые алгоритмы обработки сигналов, содержит процедуры циклической свертки, линейной свертки в общем виде, линейной свертки, использующей изоморфные переходы в ГЧС различных размерностей.

П2 Программное обеспечение основных операций с гиперкомплексными выражениями и числами

Описательная часть всех процедур данного раздела

```
> # A – гиперкомплексное число
> # B – гиперкомплексное число
> # n – размерность системы
> # Variables – набор базисных элементов
> # Mult_table – таблица умноження заданной систем
```

1. Процедура сложения Addition

```
> # Операция сложения двух гиперкомплексных чисел
> Addition:=proc(A, B, n, Variables) local i, Res;
>    Res:=0;
    for i from 1 to n do
>       Res:= Res + (coeff(A, Variables[i])+coeff(B, Variables[i]))*Variables[i];
>    od;
> return (Res);
> end proc:
```

Пример для систем третьей размерности с операндами – числами.

```
> E_seq:={seq(E[i], i=0..2)};
> A:=4*E[0]+2*E[1]-E[2];
> B:=E[0]-3*E[1]+E[2];
> C:=Addition(A,B,3,E_seq);
```

$$A := 4E_0 + 2E_1 - E_2$$
$$B := E_0 - 3E_1 + E_2$$
$$C := 5E_0 - E_1$$

Пример для систем третьей размерности с операндами – выражениями.

```
> A:=a[0]*E[0]+a[1]*E[1]-a[2]*E[2];
> B:=b[0]*E[0]-b[1]*E[1]+b[2]*E[2];
> C:=Addition(A,B,3,E_seq);
```

$$A := a_0E_0 + a_1E_1 + a_2E_2$$
$$B := b_0E_0 + b_1E_1 + b_2E_2$$
$$C := (a_0 + b_0)E_0 + (a_1 + b_1)E_1 + (a_2 + b_2)E_2$$

2. Процедура вычитания Substraction

```
> # Операция вычитания двух гиперкомплексных выражений
```

```
>Substraction:=proc(A, B, n, Variables) local i,Res;
>Res:=0;
for i from 1 to n do
>Res:= Res + (coeff(A, Variables[i])-coeff(B, Variables[i]))*Variables[i];
>od;
>return (Res);
>endproc:
```

Пример для систем третьей размерности с операндами – числами.

```
>E_seq:={seq(E[i], i=0..2)}:
> A:=4*E[0]+2*E[1]-E[2];
> B:=E[0]-3*E[1]+E[2];
> C:=Substraction(A,B,3,E_seq);
```

$$A := 4E_0 + 2E_1 - E_2$$
$$B := E_0 - 3E_1 + E_2$$
$$C := 3E_0 + 5E_1 - 2E_2$$

Пример для систем третьей размерности с операндами – выражениями.

```
> A:=a[0]*E[0]+a[1]*E[1]+a[2]*E[2]:
> B:=b[0]*E[0]+b[1]*E[1]+b[2]*E[2]:
> C:=Substraction(A,B,3,E_seq);
```

$$A := a_0 E_0 + a_1 E_1 + a_2 E_2$$
$$B := b_0 E_0 + b_1 E_1 + b_2 E_2$$
$$C := (a_0 - b_0) E_0 + (a_1 - b_1) E_1 + (a_2 - b_2) E_2$$

3. Процедура умножения Mult

```
> # Операция умножения двух гиперкомплексных чисел
>Mult:=proc(A,B,n, Variables, Mult_table) local i,j,k, Res;
>Res:=0;
for i from 1 to n do
>for j from 1 to n do
>for k from 1 to n do
>Res:= Res + coeff(A, Variables[i])*coeff(B, Variables[j])*coeff(Mult_table[i,j],
Variables[k])*Variables[k];
>od;
>od;
>od;
>return (Res);
>endproc:
```

Для примера, используем удвоение системы W в качестве гиперкомплексной числовой системы $W^{(2)}(W,W,4)$ с таблицей умножения :

$$Mult_TableWW := \begin{bmatrix} E_0 & E_1 & E_2 & E_3 \\ E_1 & E_0 & E_3 & E_2 \\ E_2 & E_3 & E_0 & E_1 \\ E_3 & E_2 & E_1 & E_0 \end{bmatrix}$$

Пример с операндами – числами.

```
>E_seq:={seq(E[i], i=0..3)}:
> A:=E[0]+E[1]-E[2]+3*E[3];
```

```
> B:=2*E[0]-3*E[1]+E[3];
> C:=Mult(A,B,4,E_seq, Mult_TableWW);
```

$$A := E_0 + E_1 - E_2 + 3E_3$$
$$B := 2E_0 - 3E_1 + E_3$$
$$C := 2E_0 - 2E_1 + 10E_3 - 10E_2$$

Пример с операндами – выражениями.

```
> A:=a[0]*E[0]+a[1]*E[1]+a[2]*E[2]+a[3]*E[3];
> B:=b[0]*E[0]+b[1]*E[1]+b[2]*E[2]+b[3]*E[3];
> C:=Mult(A,B,4,E_seq, Mult_TableWW);
```

$$A := a_0E_0 + a_1E_1 + a_2E_2 + a_3E_3$$
$$B := b_0E_0 + b_1E_1 + b_2E_2 + b_3E_3$$
$$\begin{aligned} C := {} & a_0b_0E_0 + a_0b_1E_1 + a_0b_2E_2 + a_0b_3E_3 + a_1b_0E_1 + a_1b_1E_0 \\ & + a_1b_2E_3 + a_1b_3E_2 + a_2b_0E_2 + a_2b_1E_3 + a_2b_2E_0 + a_2b_3E_1 \\ & + a_3b_0E_3 + a_3b_1E_2 + a_3b_2E_1 + a_3b_3E_0 \end{aligned}$$

4. Процедура определения единичного элемента гиперкомплексной числовой системы One_E

```
>#  Поиск единичного элемента гиперкомплексной числовой системы
> One_E:=proc(n, Variables, Mult_table) local i,j,k, M, OneEquation, X_sol, results, Res;
# строим систему уравнений, основываясь на коэффициентах при базисных элементах
>for i from 1 to n do
>OneEquation[i]:=-a[i-1];
>for j from 1 to n do
>for k from 1 to n do
        OneEquation[i]:=OneEquation[i]+a[j-1]*x[k-1]*coeff(Mult_table[j,k], Variables[i]);
>od;od;od;
>results:={seq(x[i], i=0..n-1)};
> # получаем решение системы
>X_sol:= {solve(convert(OneEquation,set),results )};
> # если решений нет – возвращаем нулевой единичный элемент
>if (nops(X_sol)=0) then
>return 0;
>endif;
>Res:=0;
> # преобразуем полученное решение в гиперкомплексное число
>for  i from 1 to n do
>Res:=Res + subs(X_sol[1],x[i-1])*Variables[i];
>od;
>returnRes;
>endproc:
```

Для примера используем ГЧС третьей размерности с заданной таблицей умножения:

```
>E_seq:={seq(E[i],i=0..2)}:
>Mult_Table3E:=array(1..3, 1..3, [[-2*E[0]-E[1],-E[1],-2*E[2]],[-E[1],E[1],0],[-
2*E[2],0,2*E[0]+E[1]]]);
>One3E:=One_E(3, E_seq, Mult_Table3E);
```

$$Mult_Table3E := \begin{bmatrix} -2E_0 - E_1 & -E_1 & -2E_2 \\ -E_1 & E_1 & 0 \\ -2E_2 & 0 & 2E_0 + E_1 \end{bmatrix}$$

$$One3E := -\frac{1}{2}E_0 + \frac{1}{2}E_1$$

Рассмотрим ГЧС с единицей в базисе – результат удвоения системы двойных чисел процедурой Грассмана – Клиффорда $W^{(2)}(W,W,4)$:

```
>E_seq:={seq(E[i],i=0..3)}:
>Mult_TableWW:= array(1..4, 1..4,
[[E[0],E[1],E[2],E[3]],[E[1],E[0],E[3],E[2]],[E[2],E[3],E[0],E[1]], [E[3],E[2],E[1],E[0]]]);
>OneWW:=One_E(4, E_seq, Mult_TableWW);
```

$$Mult_TableWW := \begin{bmatrix} E_0 & E_1 & E_2 & E_3 \\ E_1 & E_0 & E_3 & E_2 \\ E_2 & E_3 & E_0 & E_1 \\ E_3 & E_2 & E_1 & E_0 \end{bmatrix}$$

$$OneWW := E_0$$

5. Процедура определения нормы гиперкомплексного числа Norm_HNS.

```
> # Определение нормы гиперкомплексного числа
>Norm_HNS:=proc(A, n, Variables, Mult_table) local i,j,k,Norm_matrix;
>Norm_matrix:=Matrix(n,n);  # формируем матрицу для определения нормы
>for i from 1 to n do
>for j from 1 to n do
>Norm_matrix[i,j]:=0;
>for k from 1 to n do
>Norm_matrix[i,j]:=Norm_matrix[i,j]+coeff(A, Variables[k])*coeff(Mult_table[i,j],
Variables[k]);od;od;od;
>returnDeterminant(Norm_matrix);
>endproc:
```

Найдем норму числа в системе триплексных чисел в общем и частном виде.

```
>E_seq:={seq(E[i], i=0..2)};
>MultTableTriplex:=array(1..3, 1..3, [[E[0],E[1],E[2]],[E[1],0.5*(-E[0]+E[2]),-E[1]],[E[2],-
E[1],E[0]]]);
> A:=a[0]*E[0]+a[1]*E[1]+a[2]*E[2];
> N := Norm_HNS(A,3,E_seq, MultTableTriplex);
```

$$MultTableTriplex := \begin{bmatrix} E_0 & E_1 & E_2 \\ E_1 & -0.5E_0 + 0.5E_2 & -E_1 \\ E_2 & -E_1 & E_0 \end{bmatrix}$$

$$A := a_0 E_0 + a_1 E_1 + a_2 E_2$$

$$N := -0.5a_0^3 + 0.5a_0^2 a_2 - 2a_0 a_1^2 - 2a_1^2 a_2 + 0.5a_0 a_2^2 - 0.5a_2^3$$

```
> A:=-E[0]+5*E[1]+2*E[2];
> N := Norm_HNS(A,3,E_seq, MultTableTriplex);
```

$$A := -E_0 + 5E_1 + 2E_2$$

$$N := -54.5$$

6. Процедура проверки делителя нуля Nullsys_HNS

В данной процедуре производится проверка числа на то, является ли оно делителем нуля в заданной ГЧС.

```
> # Возвращает true, если заданное число является делителем нуля
>Nullsys_HNS:=proc(A, n, Variables, Mult_table);
>    # если норма числа равна нулю, возвращаем true
>if (Norm_HNS(A,n,Variables,Mult_table)=0) then
>returntrue;
>else
>returnfalse;
>endif;
>endproc:
```

Выполним проверку для некоторых чисел из системы двойных чисел. Норма системы двойных чисел $N(A)=a_0{}^2-a_1{}^2$, где $A=a_0*E_0-a_1*E_1$. Тогда для делителей нуля справедливо $a_0{}^2=a_1{}^2$. Проверим процедуру Nullsys_HNS для числа, которое удовлетворяет данному условию:

```
>E_seq:={seq(E[i], i=0..1)}:
>Mult_TableW:= array(1..2, 1..2, [[E[0],E[1]],[E[1],E[0]]]):
> A:=a[0]*E[0]-a[0]*E[1];
>Nulls:=Nullsys_HNS(A,2,E_seq,Mult_TableW);
```

$$Mult_TableW := \begin{bmatrix} E_0 & E_1 \\ E_1 & E_0 \end{bmatrix}$$

$$A := a_0 E_0 - a_0 E_1$$

$$Nulls := true$$

Проверим процедуру для числа с вещественными коэффициентами:

```
> A:=2*E[0]-E[1];
>Nulls:=Nullsys_HNS(A,2,E_seq,Mult_TableW);
```

$$A := 2 E_0 - E_1$$

$$Nulls := false$$

7. Процедура определения произведение сопряженных Conjugate_HNS

```
> # Поиск произведения сопряженных
> # normA – 0 или уже вычисленная норма числа A
>Conjugate_HNS:=proc(A, n, Variables, Mult_table, normA) local i,j,k, OneEl, N_A,
ConEquation, X_sol, results, Res;
> # находим единичный элемент системы
>OneEl:=One_E(n, Variables, Mult_table);
> # находим норму числа A
>if (normA = 0) then
>       N_A:=Norm_HNS(A, n, Variables, Mult_table);
>else
>       N_A:=normA;
>endif;
> # составляем систему уравнений
```

```
> #    _
> # A* A = N(A)*X
>for i from 1 to n do
>ConEquation[i]:=-N_A*coeff(OneEl,Variables[i]);
>for j from 1 to n do
>for k from 1 to n do
>            ConEquation[i]:=ConEquation[i]+coeff(A,Variables[j])*x[k-1]*coeff(Mult_table[j,k],
Variables[i]);od;od;od;
>results:={seq(x[i], i=0..n-1)};
>X_sol:= {solve(convert(ConEquation,set),results )};
>if (nops(X_sol)=0) then
>return 0;
>endif;
>Res:=0;
> # преобразуем полученные результаты в гиперкомплексное число
>for  i from 1 to n do
>Res:=Res + subs(X_sol[1],x[i-1])*Variables[i];od;
>returnRes;
>endproc:
```

Найдем произведение сопряженных для системы триплексных чисел.

```
>MultTableTriplex:=array(1..3, 1..3, [[E[0],E[1],E[2]],[E[1],0.5*(-E[0]+E[2]),-E[1]],[E[2],-
E[1],E[0]]]);
> A:=b[0]*E[0]+b[1]*E[1]+b[2]*E[2] ;
>E_seq:={seq(E[i], i=0..2)}:
> N:=-0.5*(b[0]^3+b[0]^2*b[2]+b[0]*b[2]^2-b[2]^3)-2*(b[0]*b[1]^2+b[1]^2*b[2]);
>A_conj:=Conjugate_HNS(A,3,E_seq,MultTableTriplex, N);
```

$$MultTableTriplex := \begin{bmatrix} E_0 & E_1 & E_2 \\ E_1 & -0.5\,E_0 + 0.5\,E_2 & -E_1 \\ E_2 & -E_1 & E_0 \end{bmatrix}$$

$$A := b_0 E_0 + b_1 E_1 + b_2 E_2$$

$$N := -0.5\,b_0^3 - 0.5\,b_0^2 b_2 - 0.5\,b_0 b_2^2 + 0.5\,b_2^3 - 2\,b_0 b_1^2 - 2\,b_1^2 b_2$$

$$\begin{aligned} A_conj := & -\left(0.2500000000\left(b_0^3 + b_0^2 b_2 + b_0 b_2^2 + 4.\,b_0 b_1^2 - 1.\,b_2^3 + 4.\right.\right. \\ & \left. b_1^2 b_2\right)\left(2.\,b_0^2 - 2.\,b_0 b_2 + b_1^2\right) E_0\Big) \Big/ \left(b_1^2 b_2 + b_0 b_1^2 - 1.\,b_0^2 b_2 + b_2^3\right. \\ & \left. + b_0^3 - 1.\,b_0 b_2^2\right) \\ & + \frac{1}{b_0^2 - 2.\,b_0 b_2 + b_1^2 + b_2^2}\Big(0.5000000000\left(b_0^3 + b_0^2 b_2 + b_0 b_2^2\right. \\ & \left. + 4.\,b_0 b_1^2 - 1.\,b_2^3 + 4.\,b_1^2 b_2\right) b_1 E_1\Big) + \Big(0.2500000000\left(-1.\right. \\ & b_1^2 b_0^3 + 7.\,b_1^2 b_0^2 b_2 - 1.\,b_1^2 b_0 b_2^2 - 7.\,b_1^2 b_2^3 - 4.\,b_1^4 b_0 - 4.\,b_1^4 b_2 \\ & + 2.\,b_2^5 + 2.\,b_0^4 b_2 - 4.\,b_0 b_2^4\Big) E_2\Big) \Big/ \left(b_1^2 b_2 + b_0 b_1^2 - 1.\,b_0^2 b_2 + b_2^3\right. \\ & \left. + b_0^3 - 1.\,b_0 b_2^2\right) \end{aligned}$$

8. Процедура деления Devision

```
> # Деление одного гиперкомплексного числа на другое число
```

```
>Devision:=proc(A, B, n, Variables, Mult_table) localConjB, Dev, NormB;
>NormB:=Norm_HNS(B, n, Variables, Mult_table);
>ConjB:=Conjugate_HNS(B, n, Variables, Mult_table, NormB);
>Dev:=Mult(A, ConjB, n, Variables, Mult_table)/NormB;
>returnDev;
>endproc:
```

Найдем частное от деления двух чисел в системе бикомплексных чисел.

```
>E_seq:={seq(E[i], i=0..3)}:
>Mult_TableCC:= array(1..4, 1..4, [[E[0],E[1],0, 0],[E[1],-E[0],0,0],[0,0,E[2],E[3]], [0,0,E[3],-
E[2]]]);
> A:=2*E[0]+E[1]-E[2]+3*E[3]; B:=E[0]+E[1]+2*E[2]-E[3];
>Dev:=Devision(A, B, 4,E_seq,Mult_TableCC);
```

$$Mult_TableCC := \begin{bmatrix} E_0 & E_1 & 0 & 0 \\ E_1 & -E_0 & 0 & 0 \\ 0 & 0 & E_2 & E_3 \\ 0 & 0 & E_3 & -E_2 \end{bmatrix}$$

$$A := 2E_0 + E_1 - E_2 + 3E_3$$

$$B := E_0 + E_1 + 2E_2 - E_3$$

$$Dev := \frac{3}{2}E_0 - \frac{1}{2}E_1 - E_2 + E_3$$

9. Процедура перехода из одной ГЧС в изоморфную в зависимости от оператора изоморфного перехода ConvertSystem

```
> # Конвертация числа из одной системы в другую с помощью заданного  > # оператора изо-
морфного перехода
> # ConvertRules – оператор изоморфного перехода
>ConvertSystem:=proc(A, n, Variables, ConvertRules) local i, Res;
>Res:=0;
>for i from 1 to n do
>Res:=Res + coeff(A, Variables[i])*ConvertRules[i];
>od;
>returnexpand(Res);
>endproc:
```

Переведем число из двойной системы W в систему $R \oplus R$ и обратно.

```
>ConvertRulesE:=[R[0]+R[1], R[0]-R[1]];
>ConvertRulesR:=[0.5*(E[0]+E[1]),0.5*(E[0]-E[1])];
>R_seq:={R[0], R[1]}:
>E_seq:={E[0], E[1]}:
> A:=5*E[0]-E[1];
> A_r:=ConvertSystem(A, 2, E_seq, ConvertRulesE);
> A_e:=ConvertSystem(A_r, 2, R_seq, ConvertRulesR);
```

$$ConvertRulesE := [R_0 + R_1, R_0 - R_1]$$

$$ConvertRulesR := [0.5E_0 + 0.5E_1, 0.5E_0 - 0.5E_1]$$

$$A := 5E_0 - E_1$$

$$A_r := 4R_0 + 6R_1$$

$$A_e := 5.0E_0 - 1.0E_1$$

ПЗ Программное обеспечение быстрых алгоритмов цифровой обработки информации с использованием методов ГЧС

1. Линейная свертка в общем виде

В процедуре LinearConvolutionGeneral реализован алгоритм линейной свертки (4.3). Процедура будет использоваться в дальнейшем для проверки вариантов алгоритма с использованием гиперкомплексных чисел.

```
> # Линейная свертка в общем виде
> # n – число элементов в массивах
> # X – массив данных
> # Y – массив данных
>LinearConvolutionGeneral:=proc(n, X, Y) local k, m, Z;
>     Z:=array(0..2*n-2);
>for k from 0 to 2*n-2 do
>       Z[k]:=0;
>for m from 0 to n-1 do
>if (k >= m and k-m < n) then
>              Z[k]:=Z[k]+X[m]*Y[k-m];
>           end if;od;od;
>    return Z;
> end proc:
```

Делаем проверку для последовательностей из восьми элементов:

```
> X:=array(0..7,[seq(x[i], i=0..7)]);
> Y:=array(0..7,[seq(y[i], i=0..7)]);
>Conv:=LinearConvolutionGeneral(8,X,Y);
```

$$Y := ARRAY\left([0..7], \left[(0) = y_0, (1) = y_1, (2) = y_2, (3) = y_3, (4) = y_4, (5) = y_5, (6) = y_6, (7) = y_7\right]\right)$$

$$\text{"Conv="}, ARRAY\left([0..14], \left[(0) = x_0 y_0, (1) = x_0 y_1 + x_1 y_0, (2) = x_0 y_2 + x_1 y_1 + x_2 y_0, (3) = x_0 y_3 + x_1 y_2 + x_2 y_1 + x_3 y_0, (4) = x_0 y_4 + x_1 y_3 + x_2 y_2 + x_3 y_1 + x_4 y_0, (5) = x_0 y_5 + x_1 y_4 + x_2 y_3 + x_3 y_2 + x_4 y_1 + x_5 y_0, (6) = x_0 y_6 + x_1 y_5 + x_2 y_4 + x_3 y_3 + x_4 y_2 + x_5 y_1 + x_6 y_0, (7) = x_0 y_7 + x_1 y_6 + x_2 y_5 + x_3 y_4 + x_4 y_3 + x_5 y_2 + x_6 y_1 + x_7 y_0, (8) = x_1 y_7 + x_2 y_6 + x_3 y_5 + x_4 y_4 + x_5 y_3 + x_6 y_2 + x_7 y_1, (9) = x_2 y_7 + x_3 y_6 + x_4 y_5 + x_5 y_4 + x_6 y_3 + x_7 y_2, (10) = x_3 y_7 + x_4 y_6 + x_5 y_5 + x_6 y_4 + x_7 y_3, (11) = x_4 y_7 + x_5 y_6 + x_6 y_5 + x_7 y_4, (12) = x_5 y_7 + x_6 y_6 + x_7 y_5, (13) = x_6 y_7 + x_7 y_6, (14) = x_7 y_7\right]\right)$$

2. Процедура выполнения циклической свертки двух последовательностей длины 2 с использованием ГЧС

Циклическая свертка для массивов из двух элементов, которая использует изоморфные переходы из двойной системы в систему W в систему $R \oplus R$ и обратно, реализована в процедуре CycleConvolution2, приведенной ниже:

```
> # Циклическая свертка числовых массивов из двух элементов с помощью изоморфных переходов
> # A – массив из двух элементов
> # B – массив из двух элементов
> # multtableCalc – таблица умножения гиперкомплексной системы 2-й размерности, в которой производится умножение
> # convertTableInitToCalc – правила изоморфного перехода из системы двойных чисел с изоморфную ей
> # convertTableCalcToInit – правила изоморфного перехода из изоморфной системы в исходную систему
> CycleConvolution2:=proc (A,B, multtableCalc, convertTableInitToCalc, convertTableCalcToInit)
local A_calc, B_calc, M_calc, M_res, S_res, E_seq, R_seq;
>R_seq:={R[0], R[1]};
>E_seq:={E[0], E[1]};
>S_res:=array(0..1);
>A_calc:=ConvertSystem(A[0]*E[0]+A[1]*E[1], 2, E_seq, convertTableInitToCalc);
>B_calc:=ConvertSystem(B[0]*E[0]+B[1]*E[1], 2, E_seq, convertTableInitToCalc);
>M_calc:=Mult(A_calc, B_calc, 2, R_seq, multtableCalc);
>M_res:=ConvertSystem(M_calc, 2, R_seq, convertTableCalcToInit);
>S_res[0]:=coeff(M_res, E[0]);
>S_res[1]:=coeff(M_res, E[1]);
>    return S_res;
> end proc:
```

Делаем проверку для двух массивов из двух чисел. Входными параметрами также являются правила умножения системы $R \oplus R$, и правила перехода из системы двойных чисел в диагональную систему и наоборот.

```
> Mult_TableR2:= array(1..2, 1..2, [[R[0],0],[0,R[1]]]):
> ConvertRulesE2:=[R[0]+R[1], R[0]-R[1]];
> ConvertRulesR2:=[0.5*(E[0]+E[1]),0.5*(E[0]-E[1])];
> X:=array(0..1,[1,1]); Y:=array(0..1,[2,-1]);
>S:=CycleConvolution2(X,Y,Mult_TableR2, ConvertRulesE2, ConvertRulesR2)
```

$$Mult_TableR2 := \begin{bmatrix} R_0 & 0 \\ 0 & R_1 \end{bmatrix}$$

$$ConvertRulesE2 := [R_0 + R_1, R_0 - R_1]$$

$$ConvertRulesR2 := [0.5\,E_0 + 0.5\,E_1, 0.5\,E_0 - 0.5\,E_1]$$

$$X := ARRAY([0..1], [(0) = 1, (1) = 1])$$

$$Y := ARRAY([0..1], [(0) = 2, (1) = -1])$$

$$"S=", ARRAY([0..1], [(0) = 1.0, (1) = 1.0])$$

3. Процедура выполнения линейной свертки двух числовых последовательностей длиной 2 с помощью ГЧС

Линейная свертка двух числовых последовательностей из двух элементов может быть выполнена с помощью процедуры LinearConvolution2, которая в свою очередь использует процедуру циклической свертки CycleConvolution2, но с представлением второго массива в инверсном виде.

```
> # Линейная свертка двух массивов из двух элементов с помощью изоморфных переходов
> # A – массив из двух элементов
> # B – массив из двух элементов
> # multtableCalc – таблица умножения гиперкомплексной системы 2-й размерности, в которой производится умножение
> # convertTableInitToCalc – правила изоморфного перехода из системы двойных чисел с изоморфную ей
> # convertTableCalcToInit – правила изоморфного перехода из изоморфной системы в исходную систему
> LinearConvolution2:=proc (A,B, multtableCalc, convertTableInitToCalc,
convertTableCalcToInit) local B_inv,A_calc, B_calc, S_calc, S_res;
>B_inv:=array(0..1,[B[1],B[0]]);
>S_calc:=array(0..1);
>S_res:=array(0..2);
# Используем циклическую свертку для размерности 2, при этом представляя второй массив в инверсном виде
>S_calc:=CycleConvolution2(A,B_inv,multtableCalc,convertTableInitToCalc,
convertTableCalcToInit);
>S_res[0]:=A[0]*B[0];
>S_res[1]:=S_calc[0];
>S_res[2]:=S_calc[1]-S_res[0];
>    return S_res;
> end proc:
```

Произведем линейную свертку двух массивов (L2) и сделаем проверку (L_gen).

```
> Mult_TableR2:= array(1..2, 1..2, [[R[0],0],[0,R[1]]]);
> ConvertRulesE2:=[R[0]+R[1], R[0]-R[1]];
> ConvertRulesR2:=[0.5*(E[0]+E[1]),0.5*(E[0]-E[1])];
> X:=array(0..1,[1,1]); Y:=array(0..1,[2,-1]);
> L2:=LinearConvolution2(X,Y,Mult_TableR2, ConvertRulesE2, ConvertRulesR2);
>L_gen:=LinearConvolutionGeneral(2,X,Y);
```

$$Mult_TableR2 := \begin{bmatrix} R_0 & 0 \\ 0 & R_1 \end{bmatrix}$$

$$ConvertRulesE := [R_0 + R_1, R_0 - R_1]$$

$$ConvertRulesR := [0.5\,E_0 + 0.5\,E_1, 0.5\,E_0 - 0.5\,E_1]$$

$$X := ARRAY([0..1], [(0) = 1, (1) = 1])$$

$$Y := ARRAY([0..1], [(0) = 2, (1) = -1])$$

$$"L2=", ARRAY([0..2], [(0) = 2, (1) = 1.0, (2) = -1.0])$$

$$"L_gen:=", ARRAY([0..2], [(0) = 2, (1) = 1, (2) = -1])$$

4. Процедура выполнения линейной свертки двух числовых последовательностей длиной 4 с помощью ГЧС

Алгоритм линейной свертки двух числовых последовательностей из четырех элементов реализован в процедуре LinearConvolution4. Для свертки используется представление

массивов данных в системе $W^{(2)}(W,W,4)$, и вычисление произведения двух чисел с переходом в систему прямой суммы вещественных чисел 4-й размерности и обратно.

```
> # Линейная свертка двух массивов из четырех элементов с помощью изоморфных переходов
> # A – массив из четырех элементов
> # B – массив из четырехэлементов
> # multtableCalc – таблица умножения гиперкомплексной системы 4-й размерности, в которой производится умножение
> # convertTableInitToCalc – правила изоморфного перехода из исходной системы в изоморфную ей
> # convertTableCalcToInit – правила изоморфного перехода из изоморфной системы в исходную систему
> LinearConvolution4:=proc (A,B, multtableCalc,  convertTableInitToCalc,
convertTableCalcToInit ) local A_R, B_R, S_res, M_E, M_R, E_seq, R_seq, A_E, B_E, A1B1,
A0B2_B0A2, A2B2;
>    A_E:=A[0]*E[0]+A[1]*E[1]+A[2]*E[2]+A[3]*E[3];
>    B_E:=B[3]*E[0]+B[2]*E[1]+B[1]*E[2]+B[0]*E[3];
>R_seq:={seq(R[i], i=0..3)};
>E_seq:={seq(E[i], i=0..3)};
>S_res:=array(0..6);
># Перемножаем гиперкомплексные числа
>    A_R:=ConvertSystem(A_E, 4, E_seq, convertTableInitToCalc);
>    B_R:=ConvertSystem(B_E, 4, E_seq, convertTableInitToCalc);
>    M_R:=Mult(A_R, B_R, 4, R_seq, multtableCalc);
>    M_E:=ConvertSystem(M_R, 4, R_seq, convertTableCalcToInit);
># Вычисляем промежуточные значения
>    A1B1:=A[1]*B[1];
>    A0B2_B0A2:=A[0]*B[2]+A[2]*B[0];
>    A2B2:=A[2]*B[2];
>S_res[0]:=A[0]*B[0];
>S_res[1]:=A[0]*B[1]+A[1]*B[0];
>S_res[2]:=A0B2_B0A2+A1B1;
>S_res[3]:=coeff(M_E,E[0]);
>S_res[4]:=coeff(M_E,E[1])-A0B2_B0A2+A2B2;
>S_res[5]:=coeff(M_E,E[2])-S_res[1];
>S_res[6]:=coeff(M_E,E[3])-S_res[0]-A1B1-A2B2;
>    return S_res;
> end proc:
```

Произведем линейную свертку двух массивов $\{1,1,2,-1\}$ и $\{1,1,2,-1\}$ с помощью процедуры LinearConvolution4 и LinearConvolutionGeneral для проверки результата.

```
> Mult_TableR4:= array(1..4, 1..4, [[R[0],0,0,0],[0,R[1],0,0], [0,0,R[2],0], [0,0,0,R[3]]]);
> ConvertRulesE4:=[R[0]+R[1]+R[2]+R[3], R[0]-R[1]+R[2]-R[3], R[0]+R[1]-R[2]-R[3], R[0]-
R[1]-R[2]+R[3]];
> ConvertRulesR4:=[0.25*(E[0]+E[1]+E[2]+E[3]),0.25*(E[0]-E[1]+E[2]-E[3]), 0.25*(E[0]+E[1]-
E[2]-E[3]), 0.25*(E[0]-E[1]-E[2]+E[3])];
> A4:=array(0..3,[1,1,2,-1]);
> B4:=array(0..3,[2,-1,1,-2]);
```

>L4 := LinearConvolution4(A4, B4, Mult_TableR4, ConvertRulesE4, ConvertRulesR4);
>L_gen:=LinearConvolutionGeneral(4,A4,B4);

$$Mult_TableR4 := \begin{bmatrix} R_0 & 0 & 0 & 0 \\ 0 & R_1 & 0 & 0 \\ 0 & 0 & R_2 & 0 \\ 0 & 0 & 0 & R_3 \end{bmatrix}$$

$$ConvertRulesE4 := [R_0 + R_1 + R_2 + R_3, R_0 - R_1 + R_2 - R_3, R_0 + R_1 - R_2 - R_3, R_0 - R_1 - R_2 + R_3]$$

$$ConvertRulesR4 := [0.25\,E_0 + 0.25\,E_1 + 0.25\,E_2 + 0.25\,E_3, 0.25\,E_0 - 0.25\,E_1 + 0.25\,E_2 - 0.25\,E_3, 0.25\,E_0 + 0.25\,E_1 - 0.25\,E_2 - 0.25\,E_3, 0.25\,E_0 - 0.25\,E_1 - 0.25\,E_2 + 0.25\,E_3]$$

$$A4 := ARRAY([0..3], [(0) = 1, (1) = 1, (2) = 2, (3) = -1])$$

$$B4 := ARRAY([0..3], [(0) = 2, (1) = -1, (2) = 1, (3) = -2])$$

"L4:=", $ARRAY([0..6], [(0) = 2, (1) = 1, (2) = 4, (3) = -5.00, (4) = 1.00, (5) = -5.00, (6) = 2.00])$

"L_gen:=", $ARRAY([0..6], [(0) = 2, (1) = 1, (2) = 4, (3) = -5, (4) = 1, (5) = -5, (6) = 2])$

5. Процедура выполнения линейной свертки двух числовых последовательностей длиной 8 с помощью ГЧС

Линейную свертку двух массивов из восьми элементов строим аналогично предыдущему пункту, но с представлением данных в системе $W^{(3)}(e,8)$.

```
> # Линейная свертка двух массивов из восьми элементов с помощью изоморфных переходов
> # A – массив из восьми элементов
> # B – массив из восьми элементов
> # multtableCalc – таблица умножения гиперкомплексной системы 8-й размерности, в которой производится умножение
> # convertTableInitToCalc – правила изоморфного перехода из исходной системы в изоморфную ей
> # convertTableCalcToInit – правила изоморфного перехода из изоморфной системы в исходную систему
> LinearConvolution8:=proc (A,B, multtableCalc,  convertTableInitToCalc,
convertTableCalcToInit ) local A_E, B_E, M_E, M_R, A_R, B_R, S_res, E_seq, R_seq, A1B1,
A0B2, A0B4, A2B2, A1B3, A0B5_A1B4, A2B3, A0B6_A2B4, A3B3, A1B5, A2B6, A4B4,
A4B5, A5B5, A4B6, A6B6;
>
A_E:=A[0]*E[0]+A[1]*E[1]+A[2]*E[2]+A[3]*E[3]+A[4]*E[4]+A[5]*E[5]+A[6]*E[6]+A[7]*E[7]
;
B_E:=B[7]*E[0]+B[6]*E[1]+B[5]*E[2]+B[4]*E[3]+B[3]*E[4]+B[2]*E[5]+B[1]*E[6]+B[0]*E[7];
>R_seq:={seq(R[i], i=0..7)};
>E_seq:={seq(E[i], i=0..7)};
>S_res:=array(0..14);
>   # Перемножаем гиперкомплексные числа
```

```
>    A_R:=ConvertSystem(A_E, 8, E_seq, convertTableInitToCalc);
>    B_R:=ConvertSystem(B_E, 8, E_seq, convertTableInitToCalc);
>    M_R:=Mult(A_R, B_R, 8, R_seq, multtableCalc);
>    M_E:=ConvertSystem(M_R, 8, R_seq, convertTableCalcToInit);
# Вычисляем промежуточные значения
>    A1B1:=A[1]*B[1];
>    A0B2:=A[0]*B[2]+A[2]*B[0];
>    A0B4:=A[0]*B[4]+A[4]*B[0];
>    A2B2:=A[2]*B[2];
>    A1B3:=A[1]*B[3]+A[3]*B[1];
>    A0B5_A1B4:=A[0]*B[5]+A[1]*B[4]+A[4]*B[1]+A[5]*B[0];
>    A2B3:=A[2]*B[3]+A[3]*B[2];
>    A0B6_A2B4:=A[0]*B[6]+A[2]*B[4]+A[4]*B[2]+A[6]*B[0];
>    A3B3:=A[3]*B[3];
>    A1B5:=A[1]*B[5]+A[5]*B[1];
>    A2B6:=A[2]*B[6]+A[6]*B[2];
>    A4B4:=A[4]*B[4];
>    A4B5:=A[4]*B[5]+A[5]*B[4];
>    A5B5:=A[5]*B[5];
>    A4B6:=A[4]*B[6]+A[6]*B[4];
>    A6B6:=A[6]*B[6];
>S_res[0]:=A[0]*B[0];
>S_res[1]:=A[0]*B[1]+A[1]*B[0];
>S_res[2]:=A0B2+A1B1;
>S_res[3]:=A[0]*B[3]+A[1]*B[2]+A[2]*B[1]+A[3]*B[0];
>S_res[4]:=A0B4+A1B3+A2B2;
>S_res[5]:=A0B5_A1B4+A2B3;
>S_res[6]:=A0B6_A2B4+A1B5+A3B3;
>S_res[7]:=coeff(M_E, E[0]);
>S_res[8]:=coeff(M_E, E[1])-A0B6_A2B4+A2B6+A4B4;
>S_res[9]:=coeff(M_E, E[2])-A0B5_A1B4+A4B5;
>S_res[10]:=coeff(M_E, E[3])-A0B4-A2B6-A1B5+A4B6+A5B5;
>S_res[11]:=coeff(M_E, E[4])-S_res[3];
>S_res[12]:=coeff(M_E, E[5])-A0B2-A1B3-A4B6+A6B6;
>S_res[13]:=coeff(M_E, E[6])-S_res[1]-A2B3-A4B5;
>S_res[14]:=coeff(M_E, E[7])-S_res[0]-A1B1-A2B2-A3B3-A4B4-A5B5-A6B6;
>    return S_res;
>endproc:
```

Произведем линейную свертку двух массивов $\{0,1,1,2,-1,1,1,-2\}$ и $\{1,2,-1,1,-2,3,-1,3\}$ с помощью процедуры LinearConvolution8 и LinearConvolutionGeneral для проверки результата.

```
>Mult_TableR8:= array(1..8, 1..8, [[R[0],0,0,0,0,0,0,0],[0,R[1],0,0,0,0,0,0], [0,0,R[2],0,0,0,0,0],
[0,0,0,R[3],0,0,0,0], [0,0,0,0,R[4],0,0,0], [0,0,0,0,0,R[5],0,0], [0,0,0,0,0,0,R[6],0],
[0,0,0,0,0,0,0,R[7]]]);
> ConvertRulesE8:=[R[0]+R[1]+R[2]+R[3]+R[4]+R[5]+R[6]+R[7], R[0]-R[1]+R[2]-R[3]+R[4]-
R[5]+R[6]-R[7], R[0]+R[1]-R[2]-R[3]+R[4]+R[5]-R[6]-R[7], R[0]-R[1]-R[2]+R[3]+R[4]-R[5]-
R[6]+R[7],R[0]+R[1]+R[2]+R[3]-R[4]-R[5]-R[6]-R[7], R[0]-R[1]+R[2]-R[3]-R[4]+R[5]-
R[6]+R[7], R[0]+R[1]-R[2]-R[3]-R[4]-R[5]+R[6]+R[7], R[0]-R[1]-R[2]+R[3]-R[4]+R[5]+R[6]-
R[7]];
```

> ConvertRulesR8:=[0.125*(E[0]+E[1]+E[2]+E[3]+E[4]+E[5]+E[6]+E[7]),0.125*(E[0]-E[1]+E[2]-E[3]+E[4]-E[5]+E[6]-E[7]), 0.125*(E[0]+E[1]-E[2]-E[3]+E[4]+E[5]-E[6]-E[7]), 0.125*(E[0]-E[1]-E[2]+E[3]+E[4]-E[5]-E[6]+E[7]),0.125*(E[0]+E[1]+E[2]+E[3]-E[4]-E[5]-E[6]-E[7]),0.125*(E[0]-E[1]+E[2]-E[3]-E[4]+E[5]-E[6]+E[7]), 0.125*(E[0]+E[1]-E[2]-E[3]-E[4]-E[5]+E[6]+E[7]), 0.125*(E[0]-E[1]-E[2]+E[3]-E[4]+E[5]+E[6]-E[7])];
> A8:=array(0..7,[0,1,1,2,-1,1,1,-2]);
> B8:=array(0..7,[1,2,-1,1,-2,3,-1,3]);
> L8:=LinearConvolution8(A8, B8, Mult_TableR8, ConvertRulesE8, ConvertRulesR8);
>L_gen:=LinearConvolutionGeneral(8,A8,B8);

$$Mult_TableR8 := \begin{bmatrix} R_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & R_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & R_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & R_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & R_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & R_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & R_6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & R_7 \end{bmatrix}$$

$$\begin{aligned} ConvertRulesE8 := [&R_0 + R_1 + R_2 + R_3 + R_4 + R_5 + R_6 + R_7, R_0 \\ &- R_1 + R_2 - R_3 + R_4 - R_5 + R_6 - R_7, R_0 + R_1 - R_2 - R_3 \\ &+ R_4 + R_5 - R_6 - R_7, R_0 - R_1 - R_2 + R_3 + R_4 - R_5 - R_6 \\ &+ R_7, R_0 + R_1 + R_2 + R_3 - R_4 - R_5 - R_6 - R_7, R_0 - R_1 + R_2 \\ &- R_3 - R_4 + R_5 - R_6 + R_7, R_0 + R_1 - R_2 - R_3 - R_4 - R_5 \\ &+ R_6 + R_7, R_0 - R_1 - R_2 + R_3 - R_4 + R_5 + R_6 - R_7] \end{aligned}$$

$$\begin{aligned} ConvertRulesR8 := [&0.125E_0 + 0.125E_1 + 0.125E_2 + 0.125E_3 \\ &+ 0.125E_4 + 0.125E_5 + 0.125E_6 + 0.125E_7, 0.125E_0 \\ &- 0.125E_1 + 0.125E_2 - 0.125E_3 + 0.125E_4 - 0.125E_5 \\ &+ 0.125E_6 - 0.125E_7, 0.125E_0 + 0.125E_1 - 0.125E_2 \\ &- 0.125E_3 + 0.125E_4 + 0.125E_5 - 0.125E_6 - 0.125E_7, \\ &0.125E_0 - 0.125E_1 - 0.125E_2 + 0.125E_3 + 0.125E_4 - 0.125E_5 \\ &- 0.125E_6 + 0.125E_7, 0.125E_0 + 0.125E_1 + 0.125E_2 \\ &+ 0.125E_3 - 0.125E_4 - 0.125E_5 - 0.125E_6 - 0.125E_7, \\ &0.125E_0 - 0.125E_1 + 0.125E_2 - 0.125E_3 - 0.125E_4 + 0.125E_5 \\ &- 0.125E_6 + 0.125E_7, 0.125E_0 + 0.125E_1 - 0.125E_2 \\ &- 0.125E_3 - 0.125E_4 - 0.125E_5 + 0.125E_6 + 0.125E_7, \\ &0.125E_0 - 0.125E_1 - 0.125E_2 + 0.125E_3 - 0.125E_4 + 0.125E_5 \\ &+ 0.125E_6 - 0.125E_7] \end{aligned}$$

$$\begin{aligned} A8 := ARRAY([0..7], [&(0) = 0, (1) = 1, (2) = 1, (3) = 2, (4) = -1, (5) \\ &= 1, (6) = 1, (7) = -2]) \end{aligned}$$

$$B8 := ARRAY([0..7], [(0) = 1, (1) = 2, (2) = -1, (3) = 1, (4) = -2, (5) = 3, (6) = -1, (7) = 3])$$

$$\text{"L8:="}, ARRAY([0..14], [(0) = 0, (1) = 1, (2) = 3, (3) = 3, (4) = 3, (5) = -4, (6) = 7, (7) = -4.000, (8) = 6.000, (9) = -1.000, (10) = 6.000, (11) = 3.000, (12) = -4.000, (13) = 5, (14) = -6.])$$

$$\text{"L_gen:="}, ARRAY([0..14], [(0) = 0, (1) = 1, (2) = 3, (3) = 3, (4) = 3, (5) = -4, (6) = 7, (7) = -4, (8) = 6, (9) = -1, (10) = 6, (11) = 3, (12) = -4, (13) = 5, (14) = -6])$$

6. Процедура выполнения циклической свертки двух последовательностей длины 4 с использованием ГЧС

В процедуре CycleConvolution4 реализован алгоритм циклической свертки двух числовых массивов из четырех элементов, использующий представление данных в гиперкомплексной числовой системе 4-й размерности и произведение этих чисел в изоморфной ей системе.

```
> # Циклическая свертка двух числовых массивов из четырех элементов
> # A – массив из четырех элементов
> # B – массив из четырех элементов
> # multtableCalc – таблица умножения гиперкомплексной системы 4-й размерности, в которой производится умножение
> # convertTableInitToCalc – правила изоморфного перехода из исходной системы в изоморфную ей
> # convertTableCalcToInit – правила изоморфного перехода из изоморфной системы в исходную систему
> CycleConvolution4:=proc (A,B, multtableCalc, convertTableInitToCalc, convertTableCalcToInit)
local A_E, B_E, A_R, B_R, M_R, M_E, S_res, E_seq, R_seq;
>    A_E:=A[0]*E[0]+A[1]*E[1]+A[2]*E[2]+A[3]*E[3];
>    B_E:=B[0]*E[0]+B[1]*E[1]+B[2]*E[2]+B[3]*E[3];
    R_seq:={seq(R[i], i=0..3)};
>    E_seq:={seq(E[i], i=0..3)};
>    S_res:=array(0..3);
>    A_R:=ConvertSystem(A_E, 4, E_seq, convertTableInitToCalc);
>    B_R:=ConvertSystem(B_E, 4, E_seq, convertTableInitToCalc);
>    M_R:=Mult(A_R, B_R, 4, R_seq, multtableCalc);
>    M_E:=ConvertSystem(M_R, 4, R_seq, convertTableCalcToInit);
>    S_res[0]:=coeff(M_E, E[0]);
>    S_res[1]:=A[0]*B[1]+A[2]*B[3]+A[1]*B[2]+A[3]*B[0];
>    S_res[2]:=coeff(M_E, E[2]);
>    S_res[3]:=coeff(M_E, E[1])+coeff(M_E, E[3])-S_res[1];
>    return S_res;
> end proc:
```

Произведем циклическую свертку двух массивов $\{1,1,2,-1\}$ и $\{2,-1,1,-2\}$ с помощью процедуры CycleConvolution:

```
> Mult_TableR4:= array(1..4, 1..4, [[R[0],0,0,0],[0,R[1],0,0], [0,0,R[2],0], [0,0,0,R[3]]]):
> ConvertRulesE4:=[R[0]+R[1]+R[2]+R[3], R[0]-R[1]+R[2]-R[3], R[0]+R[1]-R[2]-R[3], R[0]-R[1]-R[2]+R[3]]:
```

```
> ConvertRulesR4:=[0.25*(E[0]+E[1]+E[2]+E[3]),0.25*(E[0]-E[1]+E[2]-E[3]), 0.25*(E[0]+E[1]-
E[2]-E[3]), 0.25*(E[0]-E[1]-E[2]+E[3])]:
> A4:=array(0..3,[1,1,2,-1]);
> B4:=array(0..3,[2,-1,1,-2]);
> C4 := CycleConvolution4(A4, B4, Mult_TableR4, ConvertRulesE4, ConvertRulesR4);
```

$$A4 := ARRAY([0..3], [(0) = 1, (1) = 1, (2) = 2, (3) = -1])$$

$$B4 := ARRAY([0..3], [(0) = 2, (1) = -1, (2) = 1, (3) = -2])$$

$$\text{"C4:="}, ARRAY([0..3], [(0) = 5.00, (1) = -6, (2) = 4.00, (3) = -3.00])$$

7. Процедура выполнения циклической свертки двух последовательностей длины 8 с использованием ГЧС

Аналогично предыдущему пункту, строится алгоритм циклической свертки с использованием ГЧС для массивов из восьми элементов, используя ГЧС 8-й размерности. Алгоритм релизован в процедуре CycleConvolution8.

```
> # Циклическая свертка для  числовых массивов из восьми элементов
> # A – массив из восьми элементов
> # B – массив из восьми элементов
> # multtableCalc – таблица умножения гиперкомплексной системы 8-й размерности, в кото-
рой производится умножение
> # convertTableInitToCalc – правила изоморфного перехода из исходной системы в изо-
морфную ей
> # convertTableCalcToInit – правила изоморфного перехода из изоморфной системы в ис-
ходную систему
> CycleConvolution8:=proc (A,B, multtableCalc, convertTableInitToCalc, convertTableCalcToInit)
local A_E, B_E, A_R, B_R, M_R, M_E, S_res, E_seq, R_seq, Beta1_23, Beta4_5, Beta6, Beta78,
Beta910;
A_E:=A[0]*E[0]+A[1]*E[1]+A[2]*E[2]+A[3]*E[3]+A[4]*E[4]+A[5]*E[5]+A[6]*E[6]+A[7]*E[7]
;
B_E:=B[0]*E[0]+B[1]*E[1]+B[2]*E[2]+B[3]*E[3]+B[4]*E[4]+B[5]*E[5]+B[6]*E[6]+B[7]*E[7];
R_seq:={seq(R[i], i=0..7)};
>E_seq:={seq(E[i], i=0..7)};
>S_res:=array(0..7);
>    A_R:=ConvertSystem(A_E, 8, E_seq, convertTableInitToCalc);
>    B_R:=ConvertSystem(B_E, 8, E_seq, convertTableInitToCalc);
>    M_R:=Mult(A_R, B_R, 8, R_seq, multtableCalc);
>    M_E:=ConvertSystem(M_R, 8, R_seq, convertTableCalcToInit);
>    Beta1_23:=A[1]*B[0]+A[3]*B[2]+A[5]*B[4]+A[7]*B[6]-A[1]*B[2]-A[5]*B[6]-A[3]*B[4]-
A[7]*B[0];
>    Beta4_5:=A[2]*B[0]+A[3]*B[1]+A[6]*B[4]+A[7]*B[5]-A[2]*B[4]-A[3]*B[5]-A[6]*B[0]-
A[7]*B[1];
>    Beta6  :=A[1]*B[4]+A[3]*B[6]+A[5]*B[0]+A[7]*B[2];
>    Beta78 :=A[0]*B[3]+A[4]*B[7]+A[2]*B[5]+A[6]*B[1];
>    Beta910:=A[1]*B[6]+A[5]*B[2]+A[3]*B[0]+A[7]*B[4];
>S_res[0]:=coeff(M_E, E[0]);
>S_res[1]:=coeff(M_E, E[1])-Beta1_23;
>S_res[2]:=coeff(M_E, E[2])-Beta4_5;
```

```
>S_res[3]:=Beta6+Beta78;
>S_res[4]:=coeff(M_E, E[4]);
>S_res[5]:=coeff(M_E, E[5])-Beta6+Beta910;
>S_res[6]:=coeff(M_E, E[6])+Beta4_5;
>S_res[7]:=coeff(M_E, E[3])+coeff(M_E, E[7])+Beta1_23-Beta78-Beta910;
>    return S_res;
> end proc:
```

Произведем циклическую свертку двух массивов $\{0,1,1,2,-1,1,1,-2\}$ и $\{1,2,-1,1,-2,3,-1,3\}$.

```
>Mult_TableR8:= array(1..8, 1..8, [[R[0],0,0,0,0,0,0,0],[0,R[1],0,0,0,0,0,0], [0,0,R[2],0,0,0,0,0],
[0,0,0,R[3],0,0,0,0], [0,0,0,0,R[4],0,0,0], [0,0,0,0,0,R[5],0,0], [0,0,0,0,0,0,R[6],0],
[0,0,0,0,0,0,0,R[7]]]):
> ConvertRulesE8:=[R[0]+R[1]+R[2]+R[3]+R[4]+R[5]+R[6]+R[7], R[0]-R[1]+R[2]-R[3]+R[4]-
R[5]+R[6]-R[7], R[0]+R[1]-R[2]-R[3]+R[4]+R[5]-R[6]-R[7], R[0]-R[1]-R[2]+R[3]+R[4]-R[5]-
R[6]+R[7],R[0]+R[1]+R[2]+R[3]-R[4]-R[5]-R[6]-R[7], R[0]-R[1]+R[2]-R[3]-R[4]+R[5]-
R[6]+R[7], R[0]+R[1]-R[2]-R[3]-R[4]-R[5]+R[6]+R[7], R[0]-R[1]-R[2]+R[3]-R[4]+R[5]+R[6]-
R[7]]:
> ConvertRulesR8:=[0.125*(E[0]+E[1]+E[2]+E[3]+E[4]+E[5]+E[6]+E[7]),0.125*(E[0]-E[1]+E[2]-
E[3]+E[4]-E[5]+E[6]-E[7]), 0.125*(E[0]+E[1]-E[2]-E[3]+E[4]+E[5]-E[6]-E[7]), 0.125*(E[0]-E[1]-
E[2]+E[3]+E[4]-E[5]-E[6]+E[7]),0.125*(E[0]+E[1]+E[2]+E[3]-E[4]-E[5]-E[6]-E[7]),0.125*(E[0]-
E[1]+E[2]-E[3]-E[4]+E[5]-E[6]+E[7]), 0.125*(E[0]+E[1]-E[2]-E[3]-E[4]-E[5]+E[6]+E[7]),
0.125*(E[0]-E[1]-E[2]+E[3]-E[4]+E[5]+E[6]-E[7])]:
> A8:=array(0..7,[0,1,1,2,-1,1,1,-2]);
> B8:=array(0..7,[1,2,-1,1,-2,3,-1,3]);
>C8 := CycleConvolution8(A8, B8, Mult_TableR8, ConvertRulesE8, ConvertRulesR8);
```

$$A8 := ARRAY([0..7], [(0) = 0, (1) = 1, (2) = 1, (3) = 2, (4) = -1, (5) = 1, (6) = 1, (7) = -2])$$

$$B8 := ARRAY([0..7], [(0) = 1, (1) = 2, (2) = -1, (3) = 1, (4) = -2, (5) = 3, (6) = -1, (7) = 3])$$

$$"C8=", ARRAY([0..7], [(0) = 1.000, (1) = -7, (2) = 6.000, (3) = 1, (4) = 6.000, (5) = 6.000, (6) = 2.000, (7) = 3.000])$$

8. Процедура полной декомпозиции при длине последовательности 4

Построим процедуру линейной свертки двух последовательностей из четырех элементов ($\{x[0], x[1], x[2], x[3]\}$ и $\{y[0], y[1], y[2], y[3]\}$) с помощью полной декомпозиции задачи, которая сводится к четырем сверткам последовательностей $\{x[0], x[1]\}$ и $\{y[0], y[1]\}$, $\{x[0], x[1]\}$ и $\{y[2], y[3]\}$, $\{x[2], x[3]\}$ и $\{y[0], y[1]\}$, $\{x[2], x[3]\}$ и $\{y[2], y[3]\}$.

```
> # Линейная свертка двух массивов из четырех элементов методом полной декомпозиции
> # A – массив из четырех элементов
> # B – массив из четырех элементов
> # multtableCalc – таблица умножения гиперкомплексной системы 2-й размерности, в кото-
рой производится умножение
> # convertTableInitToCalc – правила изоморфного перехода из исходной системы в изо-
морфную ей
```

```
> # convertTableCalcToInit – правила изоморфного перехода из изоморфной системы в исходную систему
> ConvDecomposition4:=proc (A,B, multtableCalc,  convertTableInitToCalc, convertTableCalcToInit ) local Conv_A0A1_B0B1, Conv_A0A1_B2B3, Conv_A2A3_B0B1, Conv_A2A3_B2B3, S_res;
>S_res:=array(0..6);
>    Conv_A0A1_B0B1:=LinearConvolution2(array(0..1,[A[0],A[1]]), array(0..1,[B[0],B[1]]), multtableCalc,  convertTableInitToCalc, convertTableCalcToInit );
>    Conv_A0A1_B2B3:=LinearConvolution2(array(0..1,[A[0],A[1]]), array(0..1,[B[2],B[3]]), multtableCalc,  convertTableInitToCalc, convertTableCalcToInit );
>    Conv_A2A3_B0B1:=LinearConvolution2(array(0..1,[A[2],A[3]]), array(0..1,[B[0],B[1]]), multtableCalc,  convertTableInitToCalc, convertTableCalcToInit );
>    Conv_A2A3_B2B3:=LinearConvolution2(array(0..1,[A[2],A[3]]), array(0..1,[B[2],B[3]]), multtableCalc,  convertTableInitToCalc, convertTableCalcToInit );
>S_res[0]:=Conv_A0A1_B0B1[0];
>S_res[1]:=Conv_A0A1_B0B1[1];
>S_res[2]:=Conv_A0A1_B2B3[0] + Conv_A0A1_B0B1[2] + Conv_A2A3_B0B1[0];
>S_res[3]:=Conv_A0A1_B2B3[1] + Conv_A2A3_B0B1[1];
>S_res[4]:=Conv_A0A1_B2B3[2] + Conv_A2A3_B2B3[0] + Conv_A2A3_B0B1[2];
>S_res[5]:=Conv_A2A3_B2B3[1];
>S_res[6]:=Conv_A2A3_B2B3[2];
>    return S_res;
> end proc:
```

Выполним линейную свертку двух массивов методом полной декомпозиции и проверим результат с помощью процедуры LinearConvolutionGeneral.

```
> Mult_TableR2:= array(1..2, 1..2, [[R[0],0],[0,R[1]]]):
> ConvertRulesE2:=[R[0]+R[1], R[0]-R[1]]:
> ConvertRulesR2:=[0.5*(E[0]+E[1]),0.5*(E[0]-E[1])]:
> A4:=array(0..3,[1,1,2,-1]);
> B4:=array(0..3,[2,-1,1,-2]);
>D4 := ConvDecomposition4(A4, B4, Mult_TableR2, ConvertRulesE2, ConvertRulesR2);
>L_gen:=LinearConvolutionGeneral(4,A4,B4);
```

$$A4 := ARRAY([0..3], [(0) = 1, (1) = 1, (2) = 2, (3) = -1])$$

$$B4 := ARRAY([0..3], [(0) = 2, (1) = -1, (2) = 1, (3) = -2])$$

$$\text{"D4:="}, ARRAY([0..6], [(0) = 2, (1) = 1.0, (2) = 4., (3) = -5.0, (4) = 1.0, (5) = -5.0, (6) = 2.0])$$

$$\text{"L_gen:="}, ARRAY([0..6], [(0) = 2, (1) = 1, (2) = 4, (3) = -5, (4) = 1, (5) = -5, (6) = 2])$$

9. Процедура частичной декомпозиции при длине последовательности 4

Метод частичной декомпозиции реализован в процедуре ConvPartDecomposition4 и использует циклическую свертку последовательностей $\{x[0], x[i]\}$ и $\{y[0], y[i]\}$, $i = 1,2$. Полученные при этом значения произведений отсчетов и их парные суммы используются на дальнейших этапах расчета.

```
> # Линейная свертка двух массивов из четырех элементов методом частичной декомпозиции
```

```
> # A – массив из четырех элементов
> # B – массив из четырех элементов
> # multtableCalc – таблица умножения гиперкомплексной системы 4-й размерности, в которой производится умножение
> # convertTableInitToCalc – правила изоморфного перехода из исходной системы в изоморфную ей
> # convertTableCalcToInit – правила изоморфного перехода из изоморфной системы в исходную систему
> # multtableCalc2 – таблица умножения гиперкомплексной системы 2-й размерности, в которой производится умножение
> # convertTableInitToCalc2 – правила изоморфного перехода из системы 2-й размерностивизоморфную ей
> # convertTableCalcToInit2 – правила изоморфного перехода из изоморфной системы в систему 2-й размерности
> ConvPartDecomposition4:=proc (A,B, multtableCalc,  convertTableInitToCalc,
convertTableCalcToInit, multtableCalc2, convertTableInitToCalc2,  convertTableCalcToInit2)
local A_R, B_R, S_res, M_E, M_R, E_seq, R_seq, A_E, B_E, Conv_A0A2_B0B2,
Conv_A0A1_B0B1, A2B2;
>S_res:=array(0..6);
>    A_E:=A[0]*E[0]+A[1]*E[1]+A[2]*E[2]+A[3]*E[3];
>    B_E:=B[3]*E[0]+B[2]*E[1]+B[1]*E[2]+B[0]*E[3];
>R_seq:={seq(R[i], i=0..3)};
>E_seq:={seq(E[i], i=0..3)};
>   #Циклическаясвертка {A[0],A[1]} и {B[1],B[0]}; {A[0],A[2]} и {B[2],B[0]}
>    Conv_A0A1_B0B1:= CycleConvolution2(array(0..1,[A[0],A[1]]), array(0..1,[B[1],B[0]]),
multtableCalc2, convertTableInitToCalc2, convertTableCalcToInit2);
>    Conv_A0A2_B0B2:= CycleConvolution2(array(0..1,[A[0],A[2]]), array(0..1,[B[2],B[0]]),
multtableCalc2, convertTableInitToCalc2, convertTableCalcToInit2);
>    A_R:=ConvertSystem(A_E, 4, E_seq, convertTableInitToCalc);
>    B_R:=ConvertSystem(B_E, 4, E_seq, convertTableInitToCalc);
>    M_R:=Mult(A_R, B_R, 4, R_seq, multtableCalc);
>    M_E:=ConvertSystem(M_R, 4, R_seq, convertTableCalcToInit);
>S_res[0]:=A[0]*B[0];
    A2B2:=Conv_A0A2_B0B2[1]-S_res[0];
>S_res[1]:=Conv_A0A1_B0B1[0];
>S_res[2]:=Conv_A0A1_B0B1[1]-S_res[0]+Conv_A0A2_B0B2[0];
>S_res[3]:=coeff(M_E,E[0]);
>S_res[4]:=coeff(M_E,E[1])-Conv_A0A2_B0B2[0]+A2B2;
>S_res[5]:=coeff(M_E,E[2])-S_res[1];
>    S_res[6]:=coeff(M_E,E[3])-S_res[0]-(Conv_A0A1_B0B1[1]-S_res[0])-A2B2;
>    return S_res;
> end proc:
```

Выполним линейную свертку двух массивов $\{1,1,2,-1\}$ и $\{2,-1,1,-2\}$методом частичной декомпозиции и проверим результат с помощью процедуры LinearConvolutionGeneral.

```
> Mult_TableR4:= array(1..4, 1..4, [[R[0],0,0,0],[0,R[1],0,0], [0,0,R[2],0], [0,0,0,R[3]]]):
> ConvertRulesE4:=[R[0]+R[1]+R[2]+R[3], R[0]-R[1]+R[2]-R[3], R[0]+R[1]-R[2]-R[3], R[0]-
R[1]-R[2]+R[3]]:
> ConvertRulesR4:=[0.25*(E[0]+E[1]+E[2]+E[3]),0.25*(E[0]-E[1]+E[2]-E[3]), 0.25*(E[0]+E[1]-
E[2]-E[3]), 0.25*(E[0]-E[1]-E[2]+E[3])]:
> Mult_TableR2:= array(1..2, 1..2, [[R[0],0],[0,R[1]]]):
```

```
> ConvertRulesE2:=[R[0]+R[1], R[0]-R[1]]:
> ConvertRulesR2:=[0.5*(E[0]+E[1]),0.5*(E[0]-E[1])]:
> A4:=array(0..3,[1,1,2,-1]);
> B4:=array(0..3,[2,-1,1,-2]);
>P4 := ConvPartDecomposition4(A4, B4, Mult_TableR4, ConvertRulesE4, ConvertRulesR4,
Mult_TableR2, ConvertRulesE2, ConvertRulesR2);
>L_gen:=LinearConvolutionGeneral(4,A4,B4);
```

$$A4 := ARRAY([0..3], [(0) = 1, (1) = 1, (2) = 2, (3) = -1])$$

$$B4 := ARRAY([0..3], [(0) = 2, (1) = -1, (2) = 1, (3) = -2])$$

$$"P4:=", ARRAY([0..6], [(0) = 2, (1) = 1.0, (2) = 4.0, (3) = -5.00, (4) = 1.00, (5) = -5.00, (6) = 2.00])$$

$$"L_gen", ARRAY([0..6], [(0) = 2, (1) = 1, (2) = 4, (3) = -5, (4) = 1, (5) = -5, (6) = 2])$$

10. Процедура частичной декомпозиции при длине последовательности 8

```
> # Линейная свертка двух массивов из восьми элементов методом частичной декомпозиции
> # A – массив из восьми элементов
> # B – массив из восьмиэлементов
> # multtableCalc – таблица умножения гиперкомплексной системы 8-й размерности, в которой производится умножение
> # convertTableInitToCalc – правила изоморфного перехода из исходной системы в изоморфную ей
> # convertTableCalcToInit – правила изоморфного перехода из изоморфной системы в исходную систему
> # multtableCalc2 – таблица умножения гиперкомплексной системы 2-й размерности, в которой производится умножение
> # convertTableInitToCalc2 – правила изоморфного перехода из системы 2-й размерности в изоморфную ей
> # convertTableCalcToInit2 – правила изоморфного перехода из изоморфной системы в систему 2-й размерности
> ConvPartDecomposition8:=proc (A,B, multtableCalc,  convertTableInitToCalc,
convertTableCalcToInit, multtableCalc2, convertTableInitToCalc2,  convertTableCalcToInit2)
local A_R, B_R, S_res, M_E, M_R, E_seq, R_seq, A_E, B_E, Conv_A0A6_B0B6,
Conv_A0A5_B0B5, Conv_A0A4_B0B4, Conv_A0A3_B0B3, Conv_A0A2_B0B2,
Conv_A0A1_B0B1, A0B0, A1B1, A2B2, A1B3, A1B4, A2B3, A2B4, A3B3, A1B5, A2B6, A4B4,
A4B5, A5B5, A4B6, A6B6;
A_E:=A[0]*E[0]+A[1]*E[1]+A[2]*E[2]+A[3]*E[3]+A[4]*E[4]+A[5]*E[5]+A[6]*E[6]+A[7]*E[7]
;
B_E:=B[7]*E[0]+B[6]*E[1]+B[5]*E[2]+B[4]*E[3]+B[3]*E[4]+B[2]*E[5]+B[1]*E[6]+B[0]*E[7];
>R_seq:={seq(R[i], i=0..7)};
>E_seq:={seq(E[i], i=0..7)};
>S_res:=array(0..14);
   # Цикческаясвертка {A[0],A[1]} и {B[1],B[0]}; {A[0],A[2]} и {B[2],B[0]}; {A[0],A[3]} и
{B[3],B[0]}; {A[0],A[4]} и {B[4],B[0]}; {A[0],A[5]} и {B[5],B[0]}; {A[0],A[6]} и {B[6],B[0]}
>    Conv_A0A1_B0B1:= CycleConvolution2(array(0..1,[A[0],A[1]]), array(0..1,[B[1],B[0]]),
multtableCalc2, convertTableInitToCalc2, convertTableCalcToInit2);
>    Conv_A0A2_B0B2:= CycleConvolution2(array(0..1,[A[0],A[2]]), array(0..1,[B[2],B[0]]),
multtableCalc2, convertTableInitToCalc2, convertTableCalcToInit2);
```

```
>    Conv_A0A3_B0B3:= CycleConvolution2(array(0..1,[A[0],A[3]]), array(0..1,[B[3],B[0]]),
multtableCalc2, convertTableInitToCalc2, convertTableCalcToInit2);
>    Conv_A0A4_B0B4:= CycleConvolution2(array(0..1,[A[0],A[4]]), array(0..1,[B[4],B[0]]),
multtableCalc2, convertTableInitToCalc2, convertTableCalcToInit2);
>    Conv_A0A5_B0B5:= CycleConvolution2(array(0..1,[A[0],A[5]]), array(0..1,[B[5],B[0]]),
multtableCalc2, convertTableInitToCalc2, convertTableCalcToInit2);
>    Conv_A0A6_B0B6:= CycleConvolution2(array(0..1,[A[0],A[6]]), array(0..1,[B[6],B[0]]),
multtableCalc2, convertTableInitToCalc2, convertTableCalcToInit2);
    A_R:=ConvertSystem(A_E, 8, E_seq, convertTableInitToCalc);
>    B_R:=ConvertSystem(B_E, 8, E_seq, convertTableInitToCalc);
>    M_R:=Mult(A_R, B_R, 8, R_seq, multtableCalc);
>    M_E:=ConvertSystem(M_R, 8, R_seq, convertTableCalcToInit);
>    A0B0:=A[0]*B[0];
>    A1B1:=Conv_A0A1_B0B1[1]-A0B0;
>    A2B2:=Conv_A0A2_B0B2[1]-A0B0;
>    A1B3:=A[1]*B[3]+A[3]*B[1];
>    A1B4:=A[1]*B[4]+A[4]*B[1];
>    A2B3:=A[2]*B[3]+A[3]*B[2];
>    A2B4:=A[2]*B[4]+A[4]*B[2];
>    A3B3:=Conv_A0A3_B0B3[1]-A0B0;
>    A1B5:=A[1]*B[5]+A[5]*B[1];
>    A2B6:=A[2]*B[6]+A[6]*B[2];
>    A4B4:=Conv_A0A4_B0B4[1]-A0B0;
>    A4B5:=A[4]*B[5]+A[5]*B[4];
>    A5B5:=Conv_A0A5_B0B5[1]-A0B0;
>    A4B6:=A[4]*B[6]+A[6]*B[4];
>    A6B6:=Conv_A0A6_B0B6[1]-A0B0;
>S_res[0]:=A0B0;
>S_res[1]:=Conv_A0A1_B0B1[0];
>S_res[2]:=Conv_A0A2_B0B2[0]+A1B1;
>S_res[3]:=Conv_A0A3_B0B3[0]+A[1]*B[2]+A[2]*B[1];
>S_res[4]:=Conv_A0A4_B0B4[0]+A1B3+A2B2;
>S_res[5]:=Conv_A0A5_B0B5[0]+A1B4+A2B3;
>S_res[6]:=Conv_A0A6_B0B6[0]+A2B4+A1B5+A3B3;
>S_res[7]:=coeff(M_E, E[0]);
>S_res[8]:=coeff(M_E, E[1])-Conv_A0A6_B0B6[0]-A2B4+A2B6+A4B4;
>S_res[9]:=coeff(M_E, E[2])-Conv_A0A5_B0B5[0]-A1B4+A4B5;
>S_res[10]:=coeff(M_E, E[3])-Conv_A0A4_B0B4[0]-A2B6-A1B5+A4B6+A5B5;
>S_res[11]:=coeff(M_E, E[4])-S_res[3];
>S_res[12]:=coeff(M_E, E[5])-Conv_A0A2_B0B2[0]-A1B3-A4B6+A6B6;
>S_res[13]:=coeff(M_E, E[6])-Conv_A0A1_B0B1[0]-A2B3-A4B5;
>S_res[14]:=coeff(M_E, E[7])-A0B0-A1B1-A2B2-A3B3-A4B4-A5B5-A6B6;
>    return S_res;
> end proc:
```

Произведем свертку двух массивов $\{0,1,1,2,-1,1,1,-2\}$ и $\{1,2,-1,1,-2,3,-1,3\}$ и проведем проверку с помощью

```
>Mult_TableR8:= array(1..8, 1..8, [[R[0],0,0,0,0,0,0,0],[0,R[1],0,0,0,0,0,0], [0,0,R[2],0,0,0,0,0],
[0,0,0,R[3],0,0,0,0], [0,0,0,0,R[4],0,0,0], [0,0,0,0,0,R[5],0,0], [0,0,0,0,0,0,R[6],0],
[0,0,0,0,0,0,0,R[7]]]):
```

```
> ConvertRulesE8:=[R[0]+R[1]+R[2]+R[3]+R[4]+R[5]+R[6]+R[7], R[0]-R[1]+R[2]-R[3]+R[4]-
R[5]+R[6]-R[7], R[0]+R[1]-R[2]-R[3]+R[4]+R[5]-R[6]-R[7], R[0]-R[1]-R[2]+R[3]+R[4]-R[5]-
R[6]+R[7],R[0]+R[1]+R[2]+R[3]-R[4]-R[5]-R[6]-R[7], R[0]-R[1]+R[2]-R[3]-R[4]+R[5]-
R[6]+R[7], R[0]+R[1]-R[2]-R[3]-R[4]-R[5]+R[6]+R[7], R[0]-R[1]-R[2]+R[3]-R[4]+R[5]+R[6]-
R[7]]:
> ConvertRulesR8:=[0.125*(E[0]+E[1]+E[2]+E[3]+E[4]+E[5]+E[6]+E[7]),0.125*(E[0]-E[1]+E[2]-
E[3]+E[4]-E[5]+E[6]-E[7]), 0.125*(E[0]+E[1]-E[2]-E[3]+E[4]+E[5]-E[6]-E[7]), 0.125*(E[0]-E[1]-
E[2]+E[3]+E[4]-E[5]-E[6]+E[7]),0.125*(E[0]+E[1]+E[2]+E[3]-E[4]-E[5]-E[6]-E[7]),0.125*(E[0]-
E[1]+E[2]-E[3]-E[4]+E[5]-E[6]+E[7]), 0.125*(E[0]+E[1]-E[2]-E[3]-E[4]-E[5]+E[6]+E[7]),
0.125*(E[0]-E[1]-E[2]+E[3]-E[4]+E[5]+E[6]-E[7])]:
> Mult_TableR2:= array(1..2, 1..2, [[R[0],0],[0,R[1]]]):
> ConvertRulesE2:=[R[0]+R[1], R[0]-R[1]]:
> ConvertRulesR2:=[0.5*(E[0]+E[1]),0.5*(E[0]-E[1])]:
> A8:=array(0..7,[0,1,1,2,-1,1,1,-2]);
> B8:=array(0..7,[1,2,-1,1,-2,3,-1,3]);
>P8 := ConvPartDecomposition8(A8, B8, Mult_TableR8, ConvertRulesE8, ConvertRulesR8,
Mult_TableR2, ConvertRulesE2, ConvertRulesR2);
>L_gen:=LinearConvolutionGeneral(8,A8,B8);
```

$$A8 := ARRAY([0..7], [(0) = 0, (1) = 1, (2) = 1, (3) = 2, (4) = -1, (5) = 1, (6) = 1, (7) = -2])$$

$$B8 := ARRAY([0..7], [(0) = 1, (1) = 2, (2) = -1, (3) = 1, (4) = -2, (5) = 3, (6) = -1, (7) = 3])$$

$$\text{"P8:="}, ARRAY([0..14], [(0) = 0, (1) = 1.0, (2) = 3.0, (3) = 3.0, (4) = 3.0, (5) = -4.0, (6) = 7.0, (7) = -4.000, (8) = 6.000, (9) = -1., (10) = 6.000, (11) = 3.000, (12) = -4.000, (13) = 5., (14) = -6.0])$$

$$\text{"L_gen="}, ARRAY([0..14], [(0) = 0, (1) = 1, (2) = 3, (3) = 3, (4) = 3, (5) = -4, (6) = 7, (7) = -4, (8) = 6, (9) = -1, (10) = 6, (11) = 3, (12) = -4, (13) = 5, (14) = -6])$$

11. Циклическая свертка с использованием ГЧС и сдвига последовательностей длиной 4

```
> # Циклическая свертка для числовых массивов из четырех элементов с использованием
сдвига последовательностей
> # A – массив из четырех элементов
> # B – массив из четырех элементов
> # multtableCalc – таблица умножения гиперкомплексной системы 4-й размерности, в кото-
рой производится умножение
> # convertTableInitToCalc – правила изоморфного перехода из исходной системы в изо-
морфную ей
```

```
> # convertTableCalcToInit – правила изоморфного перехода из изоморфной системы в исходную систему
> CycleOffsetConvolution4:=proc (A,B, multtableCalc, convertTableInitToCalc, convertTableCalcToInit) local A_E, B_E, A_R, B_R, M_R, M_E, S_res, E_seq, R_seq;
    R_seq:={seq(R[i], i=0..3)};
>   E_seq:={seq(E[i], i=0..3)};
>   S_res:=array(0..3);
>   A_E:=A[0]*E[0]+A[1]*E[1]+A[2]*E[2]+A[3]*E[3];
>   B_E:=B[0]*E[0]+B[1]*E[1]+B[2]*E[2]+B[3]*E[3];
>   A_R:=ConvertSystem(A_E, 4, E_seq, convertTableInitToCalc);
>   B_R:=ConvertSystem(B_E, 4, E_seq, convertTableInitToCalc);
>   M_R:=Mult(A_R, B_R, 4, R_seq, multtableCalc);
>   M_E:=ConvertSystem(M_R, 4, R_seq, convertTableCalcToInit);
>   S_res[0]:=coeff(M_E, E[0]);
>   S_res[2]:=coeff(M_E, E[2]);
    # сдвигаем исходную последовательность B
>   B_E:=B[1]*E[0]+B[2]*E[1]+B[3]*E[2]+B[0]*E[3];
>   B_R:=ConvertSystem(B_E, 4, E_seq, convertTableInitToCalc);
>   M_R:=Mult(A_R, B_R, 4, R_seq, multtableCalc);
>   M_E:=ConvertSystem(M_R, 4, R_seq, convertTableCalcToInit);
>   S_res[1]:=coeff(M_E, E[0]);
>   S_res[3]:=coeff(M_E, E[2]);
>   return S_res;
> end proc:
```

Выполним циклическую свертку двух массивов $\{1,1,2,-1\}$ и $\{2,-1,1,-2\}$с помощью процедуры CycleOffsetConvolution4.

```
> Mult_TableR4:= array(1..4, 1..4, [[R[0],0,0,0],[0,R[1],0,0], [0,0,R[2],0], [0,0,0,R[3]]]):
> ConvertRulesE4:=[R[0]+R[1]+R[2]+R[3], R[0]-R[1]+R[2]-R[3], R[0]+R[1]-R[2]-R[3], R[0]-R[1]-R[2]+R[3]]:
> ConvertRulesR4:=[0.25*(E[0]+E[1]+E[2]+E[3]),0.25*(E[0]-E[1]+E[2]-E[3]), 0.25*(E[0]+E[1]-E[2]-E[3]), 0.25*(E[0]-E[1]-E[2]+E[3])]:
> A4:=array(0..3,[1,1,2,-1]);
> B4:=array(0..3,[2,-1,1,-2]);
> CO4 := CycleOffsetConvolution4(A4, B4, Mult_TableR4, ConvertRulesE4, ConvertRulesR4);
```

$$A4 := ARRAY([0..3], [(0) = 1, (1) = 1, (2) = 2, (3) = -1])$$

$$B4 := ARRAY([0..3], [(0) = 2, (1) = -1, (2) = 1, (3) = -2])$$

$$"CO4:=", ARRAY([0..3], [(0) = 5.00, (1) = -6.00, (2) = 4.00, (3) = -3.00])$$

12. Циклическая свертка с использованием ГЧС, сдвига последовательностей и симметрий оператора изоморфизма последовательностей длиной 4

```
> # Вспомогательная процедура получения векторов X и Y с учетом правил изоморфных переходов
> XY:=proc(A, B, n, X_idx, Y_idx, convertInitToCalc, calcVariables, convertCalcToInit, initVariables) local i,X,Y;
>X:=0;
```

```
>    Y:=0;
>    for i from 1 to n do
>  X:=X+A[i-1]*coeff(convertInitToCalc[X_idx+1], calcVariables[i]);
>  Y:=Y+B[i-1]*coeff(convertCalcToInit[Y_idx+1], initVariables[i]);
>    od;
>return X*Y;
> end proc:
```

> # Циклическая свертка для числовых массивов из четырех элементов с использованием симметрии оператора изоморфизма

```
> # A – массив из четырех элементов
> # B – массив из четырех элементов
```

> # convertTableInitToCalc – правила изоморфного перехода из исходной системы в изоморфную ей

> # convertTableCalcToInit – правила изоморфного перехода из изоморфной системы в исходную систему

```
> CycleIsomorfConvolution4:=proc (A,B, convertTableInitToCalc, convertTableCalcToInit) local
X0Y0, X1Y1, X2Y2, X3Y3, X2Y3, X3Y2, X0Y0_P_X1Y1, X0Y0_M_X1Y1, X2Y2_P_X3Y3,
X2Y3_M_X3Y2, E_seq, R_seq, S_res;
>    R_seq:={seq(R[i], i=0..3)};
>    E_seq:={seq(E[i], i=0..3)};
> S_res:=array(0..3);
>    X0Y0:=XY(A,B,4,0,0,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>    X1Y1:=XY(A,B,4,1,1,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>    X2Y2:=XY(A,B,4,2,2,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>    X3Y3:=XY(A,B,4,3,3,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>    X2Y3:=XY(A,B,4,2,3,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>    X3Y2:=XY(A,B,4,3,2,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>    X0Y0_P_X1Y1:=X0Y0+X1Y1;
>    X0Y0_M_X1Y1:=X0Y0-X1Y1;
>    X2Y2_P_X3Y3:=X2Y2+X3Y3;
>    X2Y3_M_X3Y2:=X2Y3-X3Y2;
>    S_res[0]:=X0Y0_P_X1Y1+X2Y2_P_X3Y3;
>    S_res[1]:=X0Y0_M_X1Y1-X2Y3_M_X3Y2;
>    S_res[2]:=X0Y0_P_X1Y1-X2Y2_P_X3Y3;
>    S_res[3]:=X0Y0_M_X1Y1+X2Y3_M_X3Y2;
>    return S_res;
> end proc:
```

Для примера возьмем два массива чисел, которые уже использовались для алгоритма циклической свертки, приведенного ранее. Также используем правила изоморфного перехода, используемые в предыдущих алгоритмах.

```
> A4:=array(0..3,[1,1,2,-1]);
> B4:=array(0..3,[2,-1,1,-2]);
> C4 := CycleIsomorfConvolution4(A4, B4, ConvertRulesE4, ConvertRulesR4);
```

$$A4 := ARRAY([0..3], [(0) = 1, (1) = 1, (2) = 2, (3) = -1])$$

$$B4 := ARRAY([0..3], [(0) = 2, (1) = -1, (2) = 1, (3) = -2])$$

$$\text{"C4:="}, ARRAY([0..3], [(0) = 5.00, (1) = -6.00, (2) = 4.00, (3) = -3.00])$$

13. Циклическая свертка с использованием ГЧС, сдвига последовательностей и симметрий оператора изоморфизма последовательностей длиной 8

```
> # Циклическая свертка для числовых массивов из восьми элементов с использованием симметрии оператора изоморфизма
> # A – массив из восьми элементов
> # B – массив из восьми элементов
> # convertTableInitToCalc – правила изоморфного перехода из исходной системы в изоморфную ей
> # convertTableCalcToInit – правила изоморфного перехода из изоморфной системы в исходную систему
> CycleIsomorfConvolution8:=proc (A,B, convertTableInitToCalc, convertTableCalcToInit) local X0Y0, X1Y1, X2Y2, X3Y3, X4Y4, X5Y5, X6Y6, X7Y7, X2Y3, X3Y2, X4Y6, X6Y4, X4Y7, X7Y5, X0Y0_X1Y1_X2Y2_X3Y3, X4Y4_X5Y5_X6Y6_X7Y7, X0Y0_X1Y1_X2Y3_X3Y2, X0Y0_X1Y1_X2Y3_X3Y2_2, X4Y7_X5Y5_X6Y4_X7Y5, X0Y0_X1Y1_X2Y3_X3Y2_3, X4Y6_X5Y5_X6Y4_X7Y7, E_seq, R_seq, S_res;
    R_seq:={seq(R[i], i=0..7)};
>   E_seq:={seq(E[i], i=0..7)};
>   S_res:=array(0..7);
>   X0Y0:=XY(A,B,8,0,0,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>   X1Y1:=XY(A,B,8,1,1,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>   X2Y2:=XY(A,B,8,2,2,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>   X3Y3:=XY(A,B,8,3,3,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>   X4Y4:=XY(A,B,8,4,4,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>   X5Y5:=XY(A,B,8,5,5,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>   X6Y6:=XY(A,B,8,6,6,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>   X7Y7:=XY(A,B,8,7,7,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>   X2Y3:=XY(A,B,8,2,3,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>   X3Y2:=XY(A,B,8,3,2,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>   X4Y6:=XY(A,B,8,2,3,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>   X6Y4:=XY(A,B,8,3,2,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>   X4Y7:=XY(A,B,8,2,3,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>   X7Y5:=XY(A,B,8,3,2,convertTableInitToCalc, R_seq, convertTableCalcToInit, E_seq);
>   X0Y0_X1Y1_X2Y2_X3Y3:=X0Y0+X1Y1+X2Y2+X3Y3;
>   X4Y4_X5Y5_X6Y6_X7Y7:=X4Y4+X5Y5+X6Y6+X7Y7;
>   X0Y0_X1Y1_X2Y3_X3Y2:=X0Y0-X1Y1-X2Y3+X3Y2;
>   X0Y0_X1Y1_X2Y3_X3Y2_2:=X0Y0+X1Y1-X2Y3-X3Y2;
>   X4Y7_X5Y5_X6Y4_X7Y5:=X4Y7-X5Y5-X6Y4-X7Y5;
>   X0Y0_X1Y1_X2Y3_X3Y2_3:=X0Y0-X1Y1+X2Y3-X3Y2;
>   X4Y6_X5Y5_X6Y4_X7Y7:=X4Y6-X5Y5+X6Y4-X7Y7;>
>   S_res[0]:=X0Y0_X1Y1_X2Y2_X3Y3+X4Y4_X5Y5_X6Y6_X7Y7;
>   S_res[1]:=X0Y0_X1Y1_X2Y3_X3Y2+X4Y4_X5Y5_X6Y6_X7Y7;
>   S_res[2]:=X0Y0_X1Y1_X2Y3_X3Y2_2-X4Y7_X5Y5_X6Y4_X7Y5;
>   S_res[3]:=X0Y0_X1Y1_X2Y3_X3Y2_3-X4Y6_X5Y5_X6Y4_X7Y7;
>   S_res[4]:=X0Y0_X1Y1_X2Y2_X3Y3-X4Y4_X5Y5_X6Y6_X7Y7;
>   S_res[5]:=X0Y0_X1Y1_X2Y3_X3Y2-X4Y4_X5Y5_X6Y6_X7Y7;
>   S_res[6]:=X0Y0_X1Y1_X2Y3_X3Y2_2+X4Y7_X5Y5_X6Y4_X7Y5;
>   S_res[7]:=X0Y0_X1Y1_X2Y3_X3Y2_3+X4Y6_X5Y5_X6Y4_X7Y7;
```

```
>    return S_res;
> end proc:
```

В случае с n=3 также возьмем исходные данные, которые использовались в предыдущих алгоритмах.

```
> A8:=array(0..7,[0,1,1,2,-1,1,1,-2]);
> B8:=array(0..7,[1,2,-1,1,-2,3,-1,3]);
> C8 := CycleConvolution8(A8, B8, ConvertRulesE8, ConvertRulesR8);
```

$$A8 := ARRAY([0..7], [(0) = 0, (1) = 1, (2) = 1, (3) = 2, (4) = -1, (5) = 1, (6) = 1, (7) = -2])$$

$$B8 := ARRAY([0..7], [(0) = 1, (1) = 2, (2) = -1, (3) = 1, (4) = -2, (5) = 3, (6) = -1, (7) = 3])$$

$$"C8=", ARRAY([0..7], [(0) = 1.000, (1) = -7, (2) = 6.000, (3) = 1, (4) = 6.000, (5) = 6.000, (6) = 2.000, (7) = 3.000])$$

БИБЛИОГРАФИЯ

1. Синьков М.В. Конечномерные гиперкомплексные числовые системы. Основы теории. Применения / М.В. Синьков, Ю.Е. Бояринова, Я.А. Калиновский. – К.: Инфодрук, 2010. – 388с.
2. Калиновский Я.А. Высокоразмерные изоморфные гиперкомплексные числовые системы и их использование для повышения эффективности вычислений / Я.А. Калиновский, Ю.Е.Бояринова. – К.: Инфодрук, 2012. – 183с.
3. Самарский А.А. Численные методы / А.А. Самарский, А.В. Гулин. – М.: Наука, 1989. – 432с.
4. Кушнир И.А. 101×3: 101 задача на построение; 101 задача по тригонометрии; 101 задача на уравнения / И.А. Кушнир. – К.: Факт, 2007. – 432с.
5. Нейман А.Р. Теоретические основы электротехники. Часть 2/ А.Р. Нейман, П.Л. Калантаров. – М.: ГЭИ, 1959. – 414с.
6. Диткин В. А. / Интегральные преобразования и операционное исчисление./ В.А. Диткин., А. П. Прудников – М.: Физматгиз, 1961. – 527 с.
7. Хермен Г. Восстановление изображений по проекциям. Основы реконструктивной томографии / Г.Хермен; Пер. с англ. – М.: Мир, 1983. – 352 с.
8. Наттерер Ф. Математические аспекты компьютерной томографии / Ф.Наттерер; М.: Мир, 1990. – 279 с.
9. Введение в современную томографию / К.С. Терновой, М.В. Синьков, А.И. Закидальский и др.; К.: Наук. думка, 1983. – 232 с.
10. Синьков М.В., Цибульська Є.О. Комп'ютерна томографія: Основи теорії, алгоритми реконструкції. К.: Інфодрук, 2009. – 51 с.
11. Бухштаб А.А. / Теория чисел / А.А. Бухштаб. – М.: Наука, 1966. – 384с.
12. Акушский И.Л. / Машинная арифметика в остаточных классах / И.Л. Акушский, Д.И. Юдицкий. – М.: Советское радио, 1968. – 439 с.
13. Синьков М.В. Непозиционные представления в многомерных числовых системах / М.В. Синьков, Н.М. Губарени. – К.: Наук. думка, 1979. – 140 с.
14. Курош А.Г. Лекции по общей алгебре / А.Г. Курош – М.: Наука, 1973. – 400 с.
15. Кантор И.Л. Гиперкомплексные числа / И.Л. Кантор, А.С. Солодовников. – М.: Наука, 1973. – 144 с.
16. Каліновський Я.О. Розвиток методів теорії гіперкомплексних числових систем для математичного моделювання і комп’ютерних обчислень: дисертація на здобуття наукового ступеня доктора технічних наук : 01.05.02/Я.О.Каліновський – К., 2007. – 308с.
17. Синьков М.В. Представления экспонент в изоморфных гиперкомплексных числовых системах / М.В.Синьков, Я.О.Каліновський // Реєстрація, зберігання і обробка даних. – 2009. – Т. 13, № 2.– С. 27-37.
18. Калиновский Я.А. Нормализованная форма представления экспоненциальной функции в коммутативных гиперкомплексных числовых системах / Я.А.Калиновский, Ю.Е. Бояринова // Реєстрація, зберігання і обробка даних. – 2011. – Т. 13, № 4.– С. 12-22.
19. Калиновский Я.А. Изоморфизм коммутативных гиперкомплексных числовых систем и представления экспоненциальных функций в них. Я.А. Калиновский, Т.В. Синькова, М.О.Муратова // Реєстрація, зберігання і обробка даних. – 2012. – Т. 14, № 1. – С. 10-24.
20. Калиновский Я.А. Общий случай изоморфизма коммутативных ГЧС и представления экспоненциальных функций в них / Я.А. Калиновский, Ю.Е. Бояринова // Реєстрація, зберігання і обробка даних. – 2012. – Т. 14, № 3. – С.32-46.

21. Калиновский Я.А. Критерии представимости коммутативных гиперкомплексных числовых систем прямой суммой систем низших размерностей / Я.А.Калиновский // Реєстрація, зберігання і обробка даних. – 2012. – Т. 14, № 4. – С.24-36.
22. Калиновский Я.А. Некоторые необходимые и достаточные условия изоморфизма коммутативных гиперкомплексных числовых систем / Я.А. Калиновский, Ю.Е. Бояринова, Н.А. Городько // Вісник НТУУ «КПІ», Інформатика, управління та обчислювальна техніка. Вип.55. Київ «ВЕК+», 2012. – С.205-210.
23. Ландау Л.Д. Теория поля (Теоретическая физика, т. II) / Л.Д. Ландау, Е.М. Лифшиц. – М.: Физматлит, 2003. — 536 с.
24. Davenport C.M. Commutative Hypercomplex Electromagnetic Theory [Электронный ресурс]:/ C.V.Davenport; – 2008. – Режим доступа: http://home.comcast.net/~cmdaven/hyprcplx.htm
25. Степанов С.С. Релятивистский мир [Электронный песурс]:/ С.С.Степанов;-2013. – Режим доступа: http://synset.com/ru/Релятивистский_мир
26. Catoni F. Two-dimensional hypercomplex Numbers and related Trigonometries and Geometries / Catoni F., Cannata R., Catoni V., Zampetti P. // Advances in Applied Clifford Algebras. – 2004. – Vol. 14. – No. 1. – P.47-68.
27. Березин А.В. Кватернионы в релятивистской физике/ А.В. Березин, Ю.А. Курочкин, Е.А. Толкачев. – Минск: Едиториал УРСС, 2003. — 200 с.
28. Бранец В.Н. Применение кватернионов в задачах ориентации твердого тела / В.Н. Бранець, И.П. Шмыглевский.— М.: Наука, 1973. – 319 с.
29. Курош А.Г. Лекции по общей алгебре / А.Г. Курош— М.: Наука, 1973. — 400 с.
30. Онищенко С.М. Применение гиперкомплексных чисел в теории инерциальной навигации. Автономные системы / С.М.Онищенко. – К.: Наукова думка,1983. – 208 с.
31. Toyoshima H. Computationally Efficient Implementation of Hypercomplex Digital Filters / H. Toyoshima // IEICE Trans. Fundamentals. – 2002. – Aug. – E85-A, 8. – P. 1870–1876.
32. Schutte H.D. Digitalfilter zur Verarbeitung komplexer und hypercomplexer Signale / H.D. Schutte // Dissertation. Paderborn, 1991. – 100 s.
33. Schulz D., Seitz J., LustosadaCosta J.P. Widely Linear SIMO Filtering for Hypercomplex Numbers / IEEE Information Theory Workshop. – 2011. – P. 390-394.
34. Макаров О.М. Алгоритм умножения двух кватернионов / Вычислительная математика и математическая физика – 1977, – 17:6 – С. 1574–1575.
35. Росошек С.К. Быстрый алгоритм произведения двух гиперкомплексных чисел / С. К. Росошек, А.И. Литвин, Н.Е. Черняева // Вестник ТГУ. Серия: Математика. Кибернетика. Информатика. — 2000. — № 269. — С. 66–68.
36. Lufon J.C. Sur le produit de deux quaternions. G. r. Acad, sci., 1975, 280, № 10, A665-A668.
37. Блейхут Р. Быстрые алгоритмы цифровой обработки сигналов / Р. Блейхут. – М.: Мир, 1989. – 449 с.
38. Нуссбаумер Г. Быстрое преобразование Фурье и алгоритмы вычисления сверток / Г.Нуссбаумер. – М.: Радио и связь, 1985. – 248с.
39. Сергиенко А.Б. Цифровая обработка сигналов / А.Б.Сергиенко. – СПб.: Питер, 2003. – 604с.
40. Chaitin-Chatelin F., Meskauskas T., Zaoui A. Computation with Hypercomplex Numbers // GERFACS Technical Report TR/PA/00/69 // On line: http://www.gerfacs.fr (2000).
41. Сильвестров В.В. Системы чисел // Соросовский образовательный журнал. – 1998. – №8. – С. 121-127.
42. Baez J.C. The Octonions // On line: http://math/ucr.edu/home/baez/Octonions/ octonions.html (2001).
43. Chaitin-Chatelen F., Meskauskas T., Zaoui A. Geometry and Algebra. CERFACS Technical Report TR/PA/00/74 // On line: http://www.cerfacs.fr/algor/reports/ 2000/ TR-PA-00-74.ps.gz (2000).

44. Chaitin-Chatelen F. The computing power of Geometry. CERFACS Technical Report TR/PA/99/74 // On line: http://www.cerfacs.fr/algor/reports/ 2000/ TR-PA-99-74.ps.gz (1999).
45. Noether E. Hypercomplex Grossen und Darstellungstheorie / Noether E. // Mathematische Zeitschrift. – 1929. – B. 30 P. 641-692.
46. Гольденберг Л.М. Цифровая обработка сигналов / Л.М. Гольденберг, Б.Д. Матюшкин, М.Н. Поляк. – М.: Радио и связь, 1985. – 312с.
47. Беллман Р. Прикладные задачи динамического программирования / Р. Беллман, С.Дрейфус. – М.: Наука, 1965. – 460 с.
48. Робертс С. Динамическое программирование в процессах химической технологии и методы управления / С. Робертс. – М.: Мир, 1965. – 480 с.
49. Калиновский Я.А. Структура гиперкомплексного метода быстрого вычисления линейной свертки дискретных сигналов. / Я.А.Калиновский // Реєстрація, зберігання і обробка даних. – 2013. – Т. 15, № 1.– С. 31-44.
50. Salton G., Wong A., Yang C.S. A Vector Space Model for Automatic Indexing // Communications of the ACM, 1975. – 18. – № 11. – P. 613–620.
51. Lande D.V. The model of information retrieval based on the theory of hypercomplex numerical systems / Lande D.V., Kalinovskiy Ya.A., Boyarinova Yu. E. // Preprint Arxiv 1205.3031 // on-line: http://arxiv.org/abs/1205.3031.
52. Расторгуев С.П. Информационная война. Проблемы и модели. Экзистенциальная математика/Расторгуев С.П. // – М.: Гелиос АРВ, 2006. – 240 с.
53. Губанов Д.А. Модели репутации и информационного управления в социальных сетях / Губанов Д.А., Новиков Д.А., Чхартишвили А.Г // Математическая теория игр и ее приложения, 2009.– № 2. – С. 14-37.
54. Ермаков Н.С., Иващенко А.А., Новиков Д.А. Модели репутации и норм деятельности. М.: ИПУ РАН, 2005. – 67 с.
55. Ландэ Д.В. Интернетика: Навигация в сложных сетях: модели и алгоритмы /Ландэ Д.В., Снарский А.А., Безсуднов И.В.// М.: Либроком (Editorial URSS), 2009. – 264 с.
56. Ландэ Д.В. Поиск знаний в Internet / Ландэ Д.В.// М.: Диалектика-Вильямс, 2005. – 262с.
57. Новиков Д.А. Теория управления организационными системами / Новиков Д.А. // М.: Физмалит, 2007.
58. Новиков Д.А. Теория управления организационными системами / Новиков Д.А., Чхартишвили А.Г. // М.:Синтег, 2002 – 227с.
59. C. D. Manning. An Introduction to Information Retrieval / C. D. Manning, P. Raghavan, H. Schutze. // Cambridge University Press, England. – 2009. – P.544
60. A.Maeda. Query Term Disambiguation for Web Cross-Language Information Retrieval using a Search Engine // Akira Maeda, Fatiha Sadat, Masatoshi Yoshikawa, and Shunsuke UemuraQuery // on-line: http://www.dl.is.ritsumei.ac.jp/~amaeda/pub/IRAL00-e.pdf
61. Myung-Gil Jang .Using Mutual Information to Resolve Query TranslationAmbiguities and Query Term Weighting // Myung-Gil Jang, Sung Hyon Myaeng, Se Young Park // on-line:http://acl.ldc.upenn.edu/P/P99/P99-1029.pdf
62. Габидулин Э.М. Лекции по теории информации / Э.М.Габидулин, Н.И. Пилипчук // М.: МФТИ, 2007. – 214 с.
63. Д.В.Ландэ. Гиперкомплексный подход к моделированию репутации в социальных сетях // Д.В.Ландэ, Я.А.Калиновский, Ю.Е.Бояринова / Реєстрація, зберігання і оброб. даних, 2013. – N 3. – Т15. – С. 35-42

Научное издание

Национальная академия наук Украины
Институт проблем регистрации информации

Калиновский Яков Александрович
Ландэ Дмитрий Владимирович
Бояринова Юлия Евгеньевна
Хицко Яна Владимировна

Гиперкомплексные числовые системы и быстрые алгоритмы цифровой обработки информации

Подписано к печати XX.XX.XXXX
Формат А5. Бумага офсетная. Гарнитура Times New Roman.
Печать офсетная. Ус.-печат. лист. Х.Х.
Тираж 300 экз.

Напечатано «Інфодрук»
Украина, 03113, Киев,
ул. Шпака, 2,
Тел.456 68 83

Наукове видання

Національна академія наук України
Інститут проблем реєстрації інформації

Каліновський Яків Олександрович
Ланде Дмитро Володимирович
Бояринова Юлія Євгенівна
Хіцко Яна Володимирівна

Гіперкомплексні числові системи і швидкі алгоритми цифрової обробки інформації

Підписано до друку XX.XX.XXXX
Формат А5. Папір офсетний. Гарнітура Times New Roman.
Друк офсетний. Ум.-друк.арк. Х.Х.
Наклад 300 прим.

Надруковано «Інфодрук»
Україна, 03113, Київ,
вул. Шпака, 2,
тел.456 68 83