УДК 004.942

Я. А. Калиновский[1], Ю. Е. Бояринова[1,2], А. С. Сукало[1], Я. В. Хицко[2]
[1]Институт проблем регистрации информации НАН Украины
ул. Н. Шпака, 2, 03113 Киев, Украина
[2]Национальный технический университет Украины «КПИ им. Игоря Сикорского»
Проспект Победы, 37, 03113 Киев, Украина

Система гиперкомплексных операций в Maple

Рассмотрена алгоритмически-программная система гиперкомплексных операций, а также наиболее важные процедуры, входящие в эту систему, и ее структура. Приведены листинги программ и примеры их применения.

***Ключевые слова**: гиперкомплексная числовая система, процедура, компьютерная алгебра, операция, Maple.*

1. Введение

Оперирование с гиперкомплексными числами, особенно в символьной форме, вызывает значительные трудности [1, 2], связанные с их многомерностью. Поэтому для успешного оперирования с такими объектами требуются специализированные программно-алгоритмические системы. Наиболее подходящими для создания такого типа систем являются языки символьных вычислений. Тем более что во многих из них есть средства для оперирования с некоторыми гиперкомплексными числовыми системами (ГЧС). Например, с комплексными числами, кватернионами, клиффордовыми алгебрами и др.

Из существующих многочисленных языков компьютерной алгебры наиболее распространенными являются MatLab, Mathkad, Mathematica и Maple. Авторы в своей работе остановились на системе компьютерной алгебры Maple, как одной из наиболее развитых, доступных и легко осваиваемых.

2. Цель работы

Целью работы является исследование возможности создания программно-алгоритмического комплекса для повышения эффективности математического моделирования различных научно-технических задач с использованием ГЧС различных типов и научных исследований в области теории ГЧС.

3. Основные принципы построения алгоритмически-программного комплекса гиперкомплексных вычислений

Так как система компьютерной алгебры Maple позволяет создавать частные пакеты различных вычислительных процедур, то алгоритмически-программный комплекс гиперкомплексных вычислений (в дальнейшем — АПК) представляет собой пакет, имеющий свой идентификатор. АПК можно вызывать, присоединять к программе и транспортировать на другие компьютеры. Из процедур АПК можно формировать программы вычислений, используя средства алгоритмического языка Maple.

АПК может быть инсталлирован на любом компьютере с операционной системой Windows и системой компьютерных вычислений Maple не ниже 5-й версии.

АПК открыт для пополнения новыми процедурами и редактирования существующих процедур любым пользователем, владеющим Maple.

Вызов и присоединение АПК имеет вид:

read («имя устройства: \\ путь \\ имя АПК»)

with (имя АПК)

После этого будет выведен список процедур АПК, например:

[*Add, AddHNS, Conjug, ConvertA, DirSum2, DirSumN, Divis, GenIso, HNSnumber,LibHNS, ListHNS, MultiDim, Norma, Rad2, Refill, RefillHNS, SearchHNS, SqrtEq, Subtr, Trans, Unit, VizHNS, VizInA, VizLibHNS, InAdd, InConvertHNS, InMulti, NameBas, NatMulti, RenamA*].

Много внимания при разработке АПК было уделено способам и структурам представления данных. Как было отмечено выше, АПК предназначен для оперирования с данными в гиперкомплексном виде. Как известно, общий вид гиперкомплексного числа A таков:

$$A = a_1 e_1 + a_2 e_2 + \ldots + a_n e_n , \qquad (1)$$

где n — размерность ГЧС; a_i — алгебраические выражения; e_i — элементы базиса ГЧС («мнимые единицы»).

Такую форму гиперкомплексного числа будем называть натуральной. Как показывает опыт, оперирование с гиперкомплексными числами в натуральной форме довольно неудобно. Это связано с тем, что различные операции выполняются с коэффициентами при базисных элементах, которые нужно выделять и идентифицировать.

Рассмотрим для примера такую простую операцию как сложение гиперкомплексных чисел. Пусть наряду с (1) есть число

$$B = b_1 e_1 + b_2 e_2 + \ldots + b_n e_n . \qquad (2)$$

Если в рамках Maple использовать операцию сложения, то получится следующее:

$$C = A + B = a_1 e_1 + a_2 e_2 + \ldots + a_n e_n + b_1 e_1 + b_2 e_2 + \ldots + b_n e_n . \qquad (3)$$

Выражение (3) не является натуральной формой гиперкомплексного числа. В нем нужно произвести приведение подобных. В системе Maple есть команда приведения подобных — *collect*, но при ее применении необходимо указывать переменную, по которой проводится приведение подобных. Поэтому в данном слу-

чае придется n раз применить команду *collect*, указывая каждый раз, по какой переменной производится приведение подобных:

$$C = collect((collect(collect(...(A+B),e_1),...),e_n) = (a_1+b_1)e_1+...+(a_n+b_n)e_n. \quad (4)$$

Как видно, (4) — довольно громоздкая конструкция, особенно при больших размерностях ГЧС. Таких неудобств, связанных с использованием натуральной формы представления гиперкомплексного числа, очень много.

В то же время, в системе Maple имеются средства, позволяющие избавиться от этих и многих других неудобств, связанных с использованием натуральной формы представления гиперкомплексного числа. Дело в том, что в натуральной форме представления гиперкомплексного числа важны только коэффициенты при элементах базиса и их порядковый номер в изображении гиперкомплексного числа, то есть гиперкомплексное число можно представить в виде вектора. Однако векторно-матричная форма здесь не подходит ввиду того, что компоненты матрицы и вектора должны быть однотипными. В то же время в системе Maple есть такая форма представления данных как список — list — упорядоченный набор разнотипных данных.

Для оперирования с данными в формате списков в Maple существуют многочисленные команды, которые позволяют задавать список, определять длину списка, складывать два списка одинаковой длины, определять член списка по его порядковому номеру в списке, умножать все члены списка на любое выражение и др.

Представление гиперкомплексного числа в виде списка называется списочным или внутренним представлением гиперкомплексного числа. Таким образом, вместо (1) будем пользоваться представлением

$$A = [a_1, a_2, ..., a_n]. \quad (5)$$

Тогда сумма двух чисел будет определяться очень просто:

$$C = A + B = [a_1, ..., a_n] + [b_1, ..., b_n] = [a_1 + b_1, ..., a_n + b_n], \quad (6)$$

то есть приведение подобных символьных коэффициентов в соответствии с их порядковыми номерами в числах выполняется автоматическими внутренними средствами Maple. Также одной командой выполняется и умножение всех членов списка на одно и то же выражение:

$$\lambda \cdot A = [\lambda \cdot a_1, \lambda \cdot a_2, ..., \lambda \cdot a_n]. \quad (7)$$

Таким образом, представление гиперкомплексных чисел в формате списков значительно упрощает разработку алгоритмически-программных средств. Однако такое решение требует наличия в АПК процедур для взаимно-обратного преобразования натуральной и внутренней форм представления гиперкомплексных чисел. Тем более, некоторые действия целесообразно выполнять над числами в натуральной форме. В связи с этим во многих процедурах АПК предусматривается выход в форме списка из двух элементов: первый элемент — результат в списочной форме, второй — в натуральной.

Так, например, выглядит работа процедуры генерации гиперкомплексного числа восьмой размерности:

$> A = HNSnumber(8, a, e):$
$> A[1]$

$$[a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8],$$

$> A[2]$

$$a_1e_1 + a_2e_2 + a_3e_3 + a_4e_4 + a_5e_5 + a_6e_6 + a_7e_7 + a_8e_8.$$

Также оказалось целесообразным придать списочный формат и более сложным гиперкомплексным структурам. Так, таблица Кэли умножения базисных элементов представляется трехуровневой списочной структурой: верхний уровень состоит из списка строк таблицы; второй вложенный уровень — список ячеек таблицы; третий, самый нижний уровень — список структурных констант одной ячейки.

Например, таблица Кэли для обобщенных кватернионов в натуральной форме имеет вид:

$H_{\alpha\beta}$	e_1	e_2	e_3	e_4
e_1	e_1	e_2	e_3	e_4
e_2	e_2	$-\alpha e_1$	e_4	$-\alpha e_3$
e_3	e_3	$-e_4$	$-\beta e_1$	βe_2
e_4	e_4	αe_3	$-\beta e_2$	$-\alpha\beta e_1$

а в списочной форме:

$$[[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]],[[0,1,0,0],[-\alpha,0,0,0],[0,0,0,1],[0,0,-\alpha,0]],$$
$$[[0,0,1,0],[0,0,0,-1],[-\beta,0,0,0],[0,\beta,0,0]],[[0,0,0,1],[0,0,\alpha,0],[0,-\beta,0,0],[-\alpha\beta,0,0,0]]].$$

4. Структура пакета

Структурно пакет гиперкомплексных вычислений состоит из следующих подсистем:

— выполнения алгебраических операций в ГЧС;

— манипуляции с ГЧС и таблицами Кэли;

— определения алгебраических характеристик гиперкомплексных выражений;

— хранения часто употребляемых выражений;

— выполнения модульных операций с гиперкомплексными выражениями;

— визуализации и сервиса.

4.1. Подсистема выполнения алгебраических операций в ГЧС

Подсистема состоит из следующих процедур:

HNSnumber(n,Name,NameBas) — генерация гиперкомплексного числа *n*-й размерности;

inAdd(A,B) — сложение двух гиперкомплексных чисел A и B в списочном виде;

Add(A,B,dimHNS) — сложение двух гиперкомплексных чисел A и B размерности dimHNS в натуральном виде;

Subtr(A,B,dimHNS) — вычитание двух гиперкомплексных чисел A и B размерности dimHNS в натуральной форме;

inMulti(A,B,HNS) — умножение двух чисел в виде списка;

natMulti(A,B,HNS,nBas) — умножение двух гиперкомплексных чисел в натуральной форме;

Divis(A,B,nameHNS) — деление чисел в списочном виде;

Rad2(A,Name, nameBas) — извлечение квадратного корня из гиперкомплексного числа в любой форме;

SqrtEq(A,B,C,NameHNS) — решение гиперкомплексного квадратного уравнения.

4.2. Подсистема манипуляции с ГЧС и таблицами Кэли

Подсистема состоит из процедур:

inConvertHNS (M,Name) — преобразование таблицы Кэли из естественного вида в списковый (список структурных констант);

VizHNS(Spis, nam) — визуализация списка ГЧС в таблицу Кэли с данным базисом;

nameBas(A) — определение идентификатора базиса ГЧС по гиперкомплексному числу в естественном виде;

renamA(A,nam,dimHNS) — переименование идентификатора базиса в гиперкомплексном числе в естественном виде;

VizHNS(Spis, nam) — визуализация списка ГЧС в таблицу Кэли с данным базисом;

LibHNS() — хранилище таблиц Кэли ГЧС;

SearchHNS(nameHNS, nameRepos) — процедура поиска ГЧС в хранилище по ее имени;

VizInA(inA,E) — визуализация гиперкомплексного числа из списковой формы в естественную;

ConvertA(A,DimHNS) — преобразование гиперкомплексного числа из естественной формы в списковую;

Refill(Spisok, Element) — пополнение списка на один элемент;

ListHNS(DimHNS) — генерация списка-шаблона для внутреннего представления ГЧС;

nameBas(A) — определение идентификатора базиса ГЧС по гиперкомплексному числу в естественном виде;

renamA(A,nam,dimHNS) — переименование идентификатора базиса в гиперкомплексном числе в естественном виде;

RefillHNS(nameLib, nameHNS) — удаление ГЧС из хранилища;

VizLibHNS(LibHNS) — просмотр всех ГЧС в естественном виде;

Trans(M,s,t) — транспозиция строк и столбцов таблицы Кэли ГЧС;

AddHNS(Name, Table, Comment, Type) — пополнение хранилища ГЧС;

GenIzo (L,nameHNS, newBas) — генерация изоморфной ГЧС путем линейного преобразования базиса;

DirSum2(Name1,Name2) — построение прямой суммы двух ГЧС;

DirSumN (Spisok, nameBas) — построение прямой суммы нескольких ГЧС;

MultiDim(nameHNS1,nameHNS2,nameBas,markKom,nameHNS) — умножение размерности ГЧС;

SysIzo(HNS1,HNS2) — генерация системы уравнений изоморфизма двух ГЧС;

SolIzo(SysEq) — решение системы уравнений изоморфизма двух ГЧС.

4.3. Подсистема определения алгебраических характеристик гиперкомплексных выражений

Подсистема состоит из следующих процедур:

Norma(A,nameHNS) — определение псевдонормы гиперкомплексного числа в списочном виде;

Unit(nameHNS,name) — определение единичного элемента ГЧС;

Conjug(A,nameHNS,nam) — построение сопряженного числа;

Divis(A,B,nameHNS) — процедура деления;

HNSnumber(n,Name,NameBas) — процедура генерации гиперкомплексного числа.

4.4. Подсистема хранения часто употребляемых выражений

Подсистема содержит готовые формулы выполнения различных операций и вычислений для фиксированных ГЧС. Эта подсистема может пополняться и сохраняться пользователем.

4.5. Подсистема выполнения модульных операций с гиперкомплексными выражениями

Подсистема состоит из процедур построения системы остаточных классов по гиперкомплексным модулям, определения представимости гиперкомплексного числа, алгоритма Евклида для гиперкомплексных чисел и др.

4.6. Подсистема визуализации и сервиса

Подсистема состоит из следующих процедур:

VizInA(inA,E) — визуализация гиперкомплексного числа из списковой формы в естественную;

ConvertA(A,DimHNS) — преобразование гиперкомплексного числа из естественной формы в списковую;

Refill(Spisok, Element) — пополнение списка на один элемент;

ListHNS(DimHNS) — генерация списка-шаблона для внутреннего представления ГЧС;

inConvertHNS (M,Name) — преобразование таблицы Кэли из естественного вида в списковый (список структурных констант);

RefillHNS(nameLib, nameHNS) — процедура удаления ГЧС из хранилища.

Такая структура и состав комплекса гиперкомплексных символьных вычислений в среде Maple, как будет показано далее, позволяет значительно упростить процессы создания программного обеспечения для математического моделирования различных научно-технических задач.

5. Структура некоторых процедур АПК

Для того чтобы показать эффективность применения для представления данных списковых структур, рассмотрим принципы построения некоторых про-

цедур комплекса, таких как процедуры преобразования гиперкомплексного числа из естественной формы в списковую, процедура умножения чисел в списковой форме и процедура умножения в натуральной форме.

5.1. Принципы устройства процедуры преобразования гиперкомплексного числа из естественной формы в списковую

Задача заключается в том, чтобы число A в форме (1) перевести в форму (5). На первый взгляд это сделать очень просто: сделать подстановки $e_i = 1$, $i = 1...n$ с помощью команды *subs*, после чего полученное выражение $A = a_1 + a_2 + ... + a_n$ с помощью команды *convert(A,list)* превратить в (5). Однако, если в числе A один или более коэффициентов a_i равны нулю, то есть гиперкомплексное число A неполное, то получится список, длина которого меньше n, что приведет в дальнейшем к неправильным результатам.

Идея алгоритма, работающего правильно как с полными гиперкомплексными числами, так и неполными, заключается в следующем. Сначала генерируется список длиной n — заготовка результата:

$$inA = [0, 0, ..., 0]. \tag{8}$$

Потом генерируется список той же длины, состоящий из равенств $e_i = 0$:

$$sp = [e_1 = 0, ..., e_i = 0, ..., e_n = 0],$$

для каждого из значений индексов строится список таких равенств

$$sp1 = [e_1 = 0, ..., e_i = 1, ..., e_n = 0] \tag{9}$$

и выполняется подстановка в исходное число списка (9). Если в числе A компонент $a_i = 0$, то он таким войдет в список (8). Таким образом, эта подстановка выделяет коэффициент a_i, который присваивается i-му элементу списка (8).

Полный текст программы приведен ниже. Формальными параметрами процедуры являются: A — гиперкомплексное число в естественной форме, $DimHNS$ — размерность ГЧС, в которой задано число A.

```
ConvertA := proc(A, DimHNS)
    local inA, sp, i, sp1, e, A1;
    inA := [seq(0, i = 1 ..DimHNS)];
    if A <> 0 then
        A1 := subs(nameBas(A) = e, A);
        sp := [seq(e[i] = 0, i = 1 ..DimHNS)];
        for i to DimHNS do
            sp1 := sp; sp1[i] := e[i] = 1; inA[i] := subs(sp1, A1)
        end do
    end if;
    RETURN(inA)
end proc
```

5.2. Принципы устройства процедуры умножения гиперкомплексных чисел в списковой форме

Данная процедура выполняет умножение двух гиперкомплексных A и B, которые заданы в виде списков. Умножение проводится в соответствии с законами композиции системы HNS. Таким образом, процедура реализует обобщенную формулу умножения гиперкомплексных чисел [*]:

$$A \cdot B = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} a_i b_j \gamma_{ij}^{k}, \tag{10}$$

где n — размерность ГЧС HNS, γ_{ij}^{n} — структурные константы законов композиции ГЧС HNS.

В процедуре вызывается хранилище таблиц Кэли ГЧС (процедура *LibHNS()*), в которой ищется информация о заданной ГЧС HNS (*SearchHNS(HNS, LibHNS())*), содержащей трехуровневый список структурных констант ГЧС HNS. Далее выполняется формула (10). Полный текст программы приведен ниже.

```
inMulti := proc(A, B, HNS)
    local X, k, i, j, Lib, HNS1;
    Lib := LibHNS( );
    HNS1 := SearchHNS (HNS, Lib);
    X := [seq(0, i = 1 ..nops(HNS1[1]))];
    for k to nops(HNS1[1]) do
        X[k] := 0;
        for i to nops(HNS1[1]) do
            for j to nops(HNS1[1]) do
                X[k] := X[k] + A[i]*B[j]*HNS1[i, j, k]
            end do
        end do
    end do;
    RETURN(X)
end proc
```

Приведем пример вызова и работы этой процедуры. Сгенерируем два гиперкомплексных числа четвертой размерности:

$$A := HNSnumber(4, a, e)[1], \quad B := HNSnumber(4, a, e)[1],$$
$$A := [a_1, a_2, a_3, a_4], \quad B := [b_1, b_2, b_3, b_4].$$

Построим их произведение в ГЧС $Q4N$ — некоммутативном автоудвоении обобщенной системы комплексных чисел [*], таблица Кэли которой имеет вид:

$Q4N$	E_1	E_2	E_3	E_4
E_1	E_1	E_2	E_3	E_4
E_2	E_2	$pE_1 + qE_2$	E_4	$pE_3 + qE_4$
E_3	E_3	$-E_4$	$pE_1 + qE_3$	$-pE_2 - qE_4$
E_4	E_4	$-pE_3 - qE_4$	$pE_2 + qE_4$	$-p^2E_1 - pqE_2 - pqE_3 - q^2E_4$

Вызываем процедуру:

$$C := inMulti(A, B, Q4N).$$

В результате получится список из четырех компонентов результата:

$$\begin{aligned}\big[&a_1 b_1 + a_2 b_2 p + a_3 b_3 p - a_4 b_4 p^2, a_1 b_2 + a_2 b_1 + a_2 b_2 q \\ &+ a_3 b_4 p - a_4 b_3 p - a_4 b_4 p\, q, a_1 b_3 - a_2 b_4 p + a_3 b_1 \\ &+ a_3 b_3 q + a_4 b_2 p - a_4 b_4 p\, q, a_1 b_4 - a_2 b_3 - a_2 b_4 q \\ &+ a_3 b_2 + a_3 b_4 q + a_4 b_1 + a_4 b_2 q - a_4 b_3 q - a_4 b_4 q^2\big]\end{aligned}$$

5.3. Процедура умножения двух гиперкомплексных чисел в естественной форме

Процедура умножения гиперкомплексных чисел в естественной форме использует процедуру умножения в списочном виде. Сначала число преобразуется из естественной формы в списочную, потом производится умножение в списочной форме, после чего полученное произведение преобразуется из списочной формы в натуральную и визуализируется с заданным именем базиса.

Текст процедуры приводится ниже.

```
natMulti := proc(A, B, HNS, nBas)
    VizInA(inMulti(ConvertA(A, nops(SearchHNS(HNS,
    LibHNS( )))), ConvertA(B, nops(SearchHNS(HNS,
    LibHNS( )))), HNS), nBas)
end proc
```

Пусть числа третьей размерности имеют естественную форму:

$$A := a_1 e_2 + a_2 e_2 + a_3 e_3], \quad B := [b_1 e_1 + b_2 e_2 + b_3 e_3].$$

Найдем их произведение в системе триплексных чисел $T[1]$ с таблицей умножения вида

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

Тогда их произведение будет:

$$C := natMulti(A, B, T, f),$$

$$\begin{aligned}C := &\left(a_1 b_1 - \frac{1}{2} a_2 b_2 + a_3 b_3\right) f_1 + \left(a_1 b_2 + a_2 b_1 - a_2 b_3\right. \\ &\left. - a_3 b_2\right) f_2 + \left(a_1 b_3 + \frac{1}{2} a_2 b_2 + a_3 b_1\right) f_3\end{aligned}$$

Здесь изменено имя базиса с e на f.

Рассмотрим еще вызов этой процедуры в ГЧС $R \oplus C$ [*] с таблицей умножения вида

$R \oplus C$	e_1	e_2	e_3
e_1	e_1	0	0
e_2	0	e_2	e_3
e_3	0	e_3	$-e_2$

$$C := natMulti(A, B, R \oplus C, f),$$

$$C := a_1 b_1 f_1 + (a_2 b_2 - a_3 b_3) f_2 + (a_2 b_3 + a_3 b_2) f_3 \quad .$$

6. Пример решения практической задачи с помощью АПК

В данном разделе будет проведено сравнение программ для решения задачи поворота вектора с помощью кватерниона традиционными средствами и с использованием процедур АПК. Эта задача часто возникает в системах ориентации в пространстве, навигации, компьютерной анимации и многих других.

Формула поворота вектора r в пространстве с помощью кватерниона q имеет вид [7]:

$$r' = qrq^{-1}, \tag{11}$$

где

r, r' — соответственно начальные и конечные координаты поворачиваемой точки в кватернионном виде, то есть этот кватернион будет векторным;

q, q^{-1} — прямой и обратный кватернионы, определяющие ось вращения. Все умножения в (11) — кватернионные.

В выражении (11) кватернион q должен быть нормирован, то есть его норма должна быть равна единице. Если кватернион поворота имеет вид:

$$Q = a_1 e_1 + a_2 e_2 + a_3 e_3 + a_4 e_4, \tag{12}$$

то нормированный кватернион будет таким:

$$q = \frac{a_1}{\sqrt{a_1^2 + a_2^2 + a_3^2 + a_4^2}} e_1 + \frac{a_2 e_2 + a_3 e_3 + a_4 e_4}{\sqrt{a_1^2 + a_2^2 + a_3^2 + a_4^2}}. \tag{13}$$

Геометрический смысл элементов выражения (13) таков: $\theta = 2 \arccos \frac{a_1}{\sqrt{a_1^2 + a_2^2 + a_3^2 + a_4^2}}$ — угол поворота вокруг оси, направляющие косинусы которой равны:

$$\cos \theta_i = \frac{a_i}{\sqrt{a_2^2 + a_3^2 + a_4^2}}, \; i = 2, 3, 4. \tag{14}$$

Рассмотрим задачу определения координат точки, полученной последовательными поворотами вектора около двух осей: сначала около оси, определяемой кватернионом q, а потом — около оси, определяемой кватернионом p, как это показано на рисунке.

Такой сложный поворот определяется формулой

$$r' = pqrq^{-1}p^{-1}, \qquad (15)$$

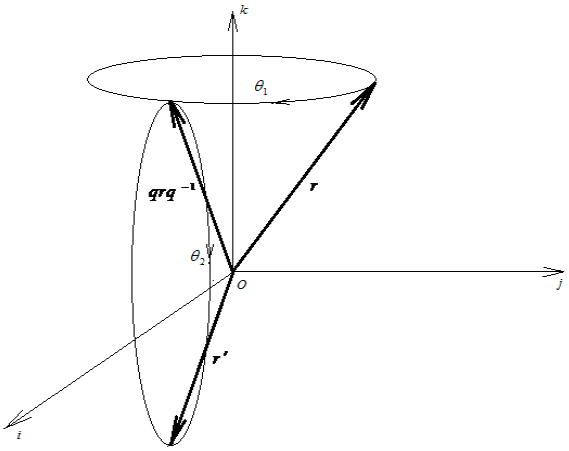
где все умножения — кватернионные [1, 7].

Обозначения в (15) такие:

r, r' — соответственно начальные и конечные координаты поворачиваемой точки в кватернионном виде, то есть этот кватернион будет векторным;

q, q^{-1} — прямой и обратный кватернионы, определяющие первую ось вращения;

p, p^{-1} — то же для второй оси вращения.



Поворот вектора в пространстве последовательно около двух осей

Таким образом, программа должна предусматривать нормирование кватернионов, определение обратных кватернионов, кватернионные перемножения по (15) и упрощение полученного результата.

Если даны исходные нормированные по (13) кватернионы q, p и r в списковом виде, а также обратные кватернионы $q^{-1} = Cq$, $p^{-1} = Cp$, то один из возможных вариантов программы в виде процедуры без применения средств АПК имеет вид:

```
Pov := proc(r, q, p, Cq, Cp)
    local AB, M1, M2, M3, M4, i, M, r1;
    AB := [a[1]*b[1] - a[2]*b[2] - a[3]*b[3] - a[4]*b[4], a
    [1]*b[2] + a[2]*b[1] + a[3]*b[4] - a[4]*b[3], a[1]*b[3
    ] - a[2]*b[4] + a[3]*b[1] + a[4]*b[2], a[1]*b[4] + a[2]
    *b[3] - a[3]*b[2] + a[4]*b[1]];
    M1 := subs(a[1] = p[1], a[2] = p[2], a[3] = p[3], a[4] = p[4], b
    [1] = q[1], b[2] = q[2], b[3] = q[3], b[4] = q[4], AB);
    M2 := subs(a[1] = M1[1], a[2] = M1[2], a[3] = M1[3], a[4]
     = M1[4], b[1] = r[1], b[2] = r[2], b[3] = r[3], b[4] = r[4], AB);
    M3 := subs(a[1] = M2[1], a[2] = M2[2], a[3] = M2[3], a[4]
     = M2[4], b[1] = Cq[1], b[2] = Cq[2], b[3] = Cq[3], b[4] = Cq
    [4], AB);
    M4 := subs(a[1] = M3[1], a[2] = M3[2], a[3] = M3[3], a[4]
     = M3[4], b[1] = Cp[1], b[2] = Cp[2], b[3] = Cp[3], b[4] = Cp
    [4], AB);
    for i to 4 do M[i] := factor(M4[i]) end do;
    r1 := [M[1], M[2], M[3], M[4]];
    RETURN(r1)
end proc
```

Как видно, основной объем этой программы занимают вычисления произведений кватернионов, которые выполнены в виде подстановок в общую формулу произведения кватернионов. Так как в АПК есть процедуры умножения гиперкомплексных чисел, то эта программа существенно упрощается. Программа с использованием средств АПК выглядит так:

$$Pov1 := \mathbf{proc}(r, q, p, Cq, Cp)\,\mathbf{for}\ i\ \mathbf{from}\ 1\ \mathbf{to}\ 4\ \mathbf{do}\ r1[i]$$
$$:= factor(inMulti(inMulti(inMulti(inMulti(p, q, H), r, H), Cq[1], H), Cp[1], H)[i])\ :\mathbf{end\ do}:\ RETURN(r1)\,\mathbf{end\ proc}$$

Результаты вычислений по обеим программам, конечно, одинаковы. Так, если решается задача поворота точки с координатами $[1,2,3]$ сначала на угол $\theta_1 = \frac{\pi}{3}$, вокруг оси, определяемой ортом $[0,0,1]$, то есть оси Oz, потом на угол $\theta_2 = \frac{\pi}{2}$ вокруг оси, определяемой ортом $[0,1,0]$, то есть оси Oy, то конечное положение точки определяется векторным кватернионом

$$r' = 3e_2 + (\sqrt{3} + 0,5)e_3 + (\sqrt{3} - 0,5)e_4 . \qquad (16)$$

Выводы

Проведенные исследования показали, что разработанные программно-алгоритмические средства охватывают большую область вычислений, связанных с моделированием процессов, описываемых гиперкомплексными числами, а также в области научных исследований.

Их использование значительно упрощает процесс разработки программного обеспечения и повышает его надежность, так как применяются многократно проверенные алгоритмы и программы. В тоже время, следует отметить, что система нуждается в дальнейшем пополнении и расширении, что является направлением научно-исследовательской работы в этой области.

1. Синьков М.В., Калиновский Я.А., Бояринова Ю.Е. Конечномерные гиперкомплексные числовые системы. Основы теории. Применения. Киев: НАН Украины, Ин-т проблем регистрации информации, 2010. 389 с.

2. Калиновский Я.А., Ландэ Д.В., Бояринова Ю.Е., Хицко Я.В. Гиперкомплексные числовые системы и быстрые алгоритмы цифровой обработки информации. Киев: Ин-т проблем регистрации информации, 2014. 130 с.

3. Von zur Gathen J., Gerhard J. Modern Computer Algebra. Cambrige: Cambrige University Press, 2013. 808 p.

4. Дьяконов В.П. Энциклопедия компьютерной алгебры; в 2-х т. Москва: ДМК-Пресс, 2009. 1264 с.

5. Kalinovsky Y.A., BoyarinovaYu.E., Sukalo A.S. Study of relations between the generalized quaternions and procedure of doubling of hypercomplex numerical systems. *Data Rec., Storage & Processing*. 2015. Vol. 17. N 1. P. 36–45.

6. Kalinovsky Y.A. The Structure of a Hyper Fast Calculation Method Linear Convolution of Discrete Signals. 2013. Vol. 15. # 1. P. 31–44.

7. Liefke H. Quaternion Calculus for Modeling Rotations in 3D Space. URL: www.liefke.com/hartmut (1998)

Поступила в редакцию 31.05.2017