УДК 517.951
ББК 22.161

РЕАЛИЗАЦИЯ МУЛЬТИСКОБКИ КРУГЛИКОВА–ЛЫЧАГИНА В СИСТЕМЕ КОМПЬЮТЕРНОЙ АЛГЕБРЫ MAPLE

Тычков С. Н.[1]

(Учреждение Российской академии наук Институт проблем управления им. В.А. Трапезникова РАН, Москва)

Описывается реализация мультискобки Кругликова–Лычагина в системе компьютерной алгебры Maple. Приводится пример использования мультискобки для исследования совместности системы дифференциальных уравнений в частных производных.

Ключевые слова: дифференциальные уравнения, мультискобка Кругликова–Лычагина, Maple.

Введение

В работе предлагается реализация мультискобки Кругликова–Лычагина, которая используется для решения вопроса о формальной интегрируемости системы дифференциальных уравнений [1]. Такие задачи часто возникают в дифференциальной геометрии, теории дифференциальных уравнений и геометрической теории управления.

Рассмотрим систему $m+1$ дифференциального уравнения в частных производных порядка s

$$(1)\quad \mathcal{E}: \left\{ F_i\left(x, u, \ldots, \frac{\partial^{|\sigma|}u}{\partial x^{\sigma}}, \ldots\right) = 0, \qquad i = 1, \ldots, m+1, \right.$$

где $x = (x_1, x_2, \ldots, x_n)$ — вектор размерности n; $u = (u^1, u^2, \ldots, u^m)$ — вектор-функция от x; $\sigma = (\sigma_1, \ldots, \sigma_n)$ — мультииндексы длины $|\sigma| = \sigma_1 + \ldots + \sigma_n$. В этих обозначениях под

[1] *Сергей Николаевич Тычков, аспирант,* (*sergey.lab06@live.ru*).

$\partial^{|\sigma|}u^i/\partial x^\sigma$ мы понимаем частную производную

$$\frac{\partial^{|\sigma|}u^i}{\partial x_1^{\sigma_1}\partial x_2^{\sigma_2}\dots\partial x_n^{\sigma_n}}.$$

Для упрощения записи введем обозначение:

$$p_\sigma^k = \frac{\partial^{|\sigma|}u^k}{\partial x^\sigma}, \qquad k = 1,\dots,m.$$

Под оператором полной производной мы понимаем оператор

$$\frac{\mathrm{d}}{\mathrm{d}x_j} = \frac{\partial}{\partial x_j} + \sum_{k,\sigma} p_{\sigma+1_j}^k \frac{\partial}{\partial p_\sigma^k},$$

где $\sigma + 1_j$ — мультииндекс, полученный из мультииндекса σ увеличением его j-й компоненты на единицу.

Рассмотрим дифференциальное уравнение порядка s вида

$$F\left(x, u, p_{\sigma_1}^1, \dots, p_{\sigma_m}^m\right) = 0. \tag{2}$$

Определение 1. *Линеаризацией [2] уравнения (2) называется вектор дифференциальных операторов*

$$\ell(F) = \left(L^1, \dots, L^m\right), \tag{3}$$

где

$$L^k = \sum_\sigma \frac{\partial F}{\partial p_\sigma^k}\frac{\mathrm{d}^{|\sigma|}}{\mathrm{d}^\sigma x}, \qquad k = 1,\dots,m,$$

$$\frac{\mathrm{d}^{|\sigma|}}{\mathrm{d}^\sigma x} = \frac{\mathrm{d}^{\sigma_1}}{\mathrm{d}^{\sigma_1}x_1} \circ \dots \circ \frac{\mathrm{d}^{\sigma_n}}{\mathrm{d}^{\sigma_n}x_n},$$

$\circ$ *— композиция дифференциальных операторов, а*

$$\frac{\mathrm{d}^{\sigma_j}}{\mathrm{d}^{\sigma_j}x_j}, \quad j = 1,\dots,n$$

— оператор полной производной порядка σ_j по переменной x_j.

Линеаризуя каждое уравнение системы (1), получим линеаризованную систему в виде:

$$\begin{pmatrix} L_1^1 & L_1^2 & \dots & L_1^m \\ \vdots & \vdots & \ddots & \vdots \\ L_m^1 & L_m^2 & \dots & L_m^m \\ L_{m+1}^1 & L_{m+1}^2 & \dots & L_{m+1}^m \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{4}$$

где L_i^j — j-я компонента вектора $\ell(F_i)$.

Пусть $\mathcal{D}_i = \left(\mathcal{D}_i^1, \dots, \mathcal{D}_i^s\right)$, $i = 1,\dots,s$, где $\mathcal{D}_i^j$ — линейный дифференциальный оператор. Множество линейных дифференциальных операторов образует некоммутативную алгебру относительно композиции.

Определение 2. *Некоммутативным определителем* $\langle \mathcal{D}_1, \dots, \mathcal{D}_m \rangle$ *называется любое (выбранное и зафиксированное) обобщение определителя для алгебры дифференциальных операторов:*

$$\langle \mathcal{D}_1, \dots, \mathcal{D}_m \rangle = \operatorname{Ndet} \begin{pmatrix} \mathcal{D}_1^1 & \mathcal{D}_1^2 & \dots & \mathcal{D}_1^m \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{D}_m^1 & \mathcal{D}_m^2 & \dots & \mathcal{D}_m^m \end{pmatrix} = \sum_{\alpha \in S_m} (-1)^\alpha \mathcal{D}^1_{\alpha(1)} \dots \mathcal{D}^m_{\alpha(m)},$$

где S_m *— множество всех перестановок из* m *элементов;* $(-1)^\alpha$ *— четность перестановки* $\alpha \in S_m$.

Впоследствии, вычисляя некоммутативный определитель, мы всегда будем раскладывать его по первому столбцу.

Определение 3. *Мультискобкой Кругликова–Лычагина функций* $F_1, F_2, \dots, F_{m+1}$ *называется функция на пространстве джетов [2]*

$$\{F_1, F_2, \dots, F_{m+1}\} \stackrel{def}{=} \sum_{i=1}^{m+1} \left\langle L_1, \dots, \widehat{L}_i, \dots, L_{m+1} \right\rangle F_i, \tag{5}$$

где $L_i = \ell(F_i) = \left(L_i^1, L_i^2, \dots, L_i^m\right)$, $i = 1, \dots, m+1$, *а* $\widehat{L}_i$ *означает пропуск* i*-й компоненты.*

Основной результат, который мы используем для исследования формальной интегрируемости системы дифференциальных уравнений (1), состоит в следующем:

1) Если система $\mathcal{E}$ формально интегрируема, то мультискобка

$$\{F_1, \dots, F_m, F_{m+1}\}$$

равна нулю в силу системы $\mathcal{E}$.

2) Если система $\mathcal{E}$ находится в полном пересечении ([1, 3]), то $\mathcal{E}$ формально интегрируема тогда и только тогда, когда мультискобка

$$\{F_1, \dots, F_m, F_{m+1}\}$$

равна нулю в силу системы $\mathcal{E}$.

Если рассматриваемая система состоит из более чем $m+1$ уравнений, то необходимо рассмотреть скобки для всевозможных наборов из $m+1$ уравнений. Подробное изложение этих результатов можно найти в [5].

1. Вспомогательные процедуры

Для реализации вычисления формулы (5) нам необходимы следуюшие процедуры:

1) операции над дифференциальными операторами: сложение, умножение на функцию и композиция;

2) вычисление некоммутативного определителя матрицы;

3) вычисление линеаризации системы дифференциальных уравнений.

1.1. Операции над дифференциальными операторами

Для представления дифференциальных операторов в программе мы выбрали определенный в *Maple* тип *table* [6], который представляет собой ассоциативный массив. Ключом в нашем массиве будет мультииндекс σ, а значением — алгебраическое выражение, стоящее перед соответствующей индексу частной производной.

Таким образом, каждая пара ключ-значение $((\sigma_1, \ldots, \sigma_n), F_\sigma(x_1, \ldots, x_n))$ представляет оператор $F_\sigma(x_1, \ldots, x_n)\,\partial^{|\sigma|}/\partial x^\sigma$. Тогда весь ассоциативный массив T представляет дифференциальный оператор

$$\sum_{\sigma \in \mathcal{K}(T)} F_\sigma(x_1, \ldots, x_n)\,\frac{\partial^{|\sigma|}}{\partial x^\sigma},$$

где $\mathcal{K}(T)$ — множество ключей массива. Далее в зависимости от контекста мы будем понимать под T и сам оператор, и ассоциативный массив, которым оператор представляется.

Сложение дифференциальных операторов. В выбранном представлении операторов сумма двух операторов

$$T_1 = \sum_{\sigma \in \mathcal{K}(T_1)} F_\sigma \frac{\partial^{|\sigma|}}{\partial x^\sigma} \quad \text{и} \quad T_2 = \sum_{\tau \in \mathcal{K}(T_2)} G_\tau \frac{\partial^{|\tau|}}{\partial x^\tau}$$

задается следующим образом:

$$T_1 + T_2 = \sum_{\sigma \in \mathcal{K}(T_1) \setminus \mathcal{K}(T_2)} F_\sigma \frac{\partial^{|\sigma|}}{\partial x^\sigma} +$$

$$+ \sum_{\tau \in \mathcal{K}(T_2) \setminus \mathcal{K}(T_1)} G_\tau \frac{\partial^{|\tau|}}{\partial x^\tau} + \sum_{\sigma \in \mathcal{K}(T_1) \cap \mathcal{K}(T_2)} (F_\sigma + G_\sigma) \frac{\partial^{|\sigma|}}{\partial x^\sigma}.$$

Формула для разности аналогична. Реализация сложения двух операторов на языке *Maple* представлена ниже.

```
diffopPlus := proc( diffop1::table, diffop2::table )::table;
local result::table, sigma::list;
  result := copy( diffop1 );
  for sigma in indices( diffop2 ) do
    if sigma in {indices( result )} then
      result[op( sigma )] := result[op( sigma )] + diffop2[op( sigma )];
    else
      result[op( sigma )] := diffop2[op( sigma )];
    end if;
  end do;
  return copy( result );
end proc;
```

Умножение дифференциального оператора на функцию. Умножение оператора

$$T = \sum_{\sigma \in \mathcal{K}(T)} F_\sigma \frac{\partial^{|\sigma|}}{\partial x^\sigma}$$

на функцию $g(x_1, \ldots, x_n)$ слева задается формулой

$$\sum_{\sigma \in \mathcal{K}(T)} g F_\sigma \frac{\partial^{|\sigma|}}{\partial x^\sigma}.$$

Код, ее реализущий:

```
diffopMult := proc( diffOp::table, g )::table;
local result::table, sigma::list;
  result := copy( diffOp );
  for sigma in indices( diffOp ) do
    result[op( sigma )] := result[op( sigma )] * g;
  end do;
  return copy( result ):
end proc:
```

Композиция дифференцирования и дифференциального монома. Пусть дан моном

$$M = f\frac{\partial^{|\tau|}}{\partial x^{\tau}},$$

где τ — мультииндекс. Рассмотрим композицию дифференцирования по переменной x_i и монома M. Формула для вычисления композиции вытекает из правила Лейбница и выглядит так:

$$\frac{\partial}{\partial x_i} \circ M = \frac{\partial f}{\partial x_i}\frac{\partial^{|\tau|}}{\partial x^{\tau}} + f\frac{\partial^{|\tau|+1}}{\partial x_1^{\tau_1}\dots\partial x_2^{\tau_i+1}\dots\partial x_n^{\tau_n}}$$

Процедура на *Maple*, вычисляющая данную композицию:

```
diffopDMonomial := proc( diffMon::table, varIndex::integer )::table;
local result::table, sigma::list, othersigma::list;
  if nops( [indices( diffMon )] ) <> 1 then
    error "Incorrect monomial has size %1, indices=%2",
      nops( [indices( diffMon )] ), [indices( diffMon )]
  end if;
  if ivarsCount < varIndex or varIndex < 1 then
    error "Incorrect variable's index %1", varIndex;
  end if;
  sigma := indices( diffMon )[1];
  othersigma := sigma;
  othersigma[varIndex] := othersigma[varIndex] + 1;
  result := copy( diffMon );
  result[op( sigma )] := TDiff( diffMon[op( sigma )], varIndex );
  result[op( othersigma )] := diffMon[op( sigma )];
  return copy( result );
end proc:
```

Композиция дифференцирования и дифференциального оператора. Используя функцию *diffopDMonomial,* реализованную выше, мы можем построить композицию оператора взятия частной производной и дифференциального оператора. Пусть задан оператор

$$T = \sum_{\sigma\in\mathcal{K}(T)} M_\sigma,$$

где $M_\sigma = F_\sigma\frac{\partial^{|\sigma|}}{\partial x^{\sigma}}$. Формула для композиции записывается следующим способом:

$$\frac{\partial}{\partial x_i} \circ T = \sum_{\sigma\in\mathcal{K}(T)} \frac{\partial}{\partial x_i} \circ M_\sigma.$$

Исходный код этой процедуры на *Maple*:

```
diffopDOperator := proc( diffOp::table, varIndex::integer )::table;
local result::table, sigma::list, diffMon::table;
  if ivarsCount < varIndex or varIndex < 1 then
    error "Incorrect variable's index %1", varIndex;
  end if;
  result := table();
  for sigma in indices( diffOp ) do
    diffMon := table( [op( sigma ) = diffOp[op( sigma )]] );
    result := diffopPlus( result, diffopDMonomial( diffMon, varIndex ) );
  end do;
  return result;
end proc:
```

Композиция дифференциального монома и дифференциального оператора. Пусть

$$N = f\frac{\partial^{|\tau|}}{\partial x^{\tau}}$$

— дифференциальный моном, а

$$T = \sum_{\sigma\in\mathcal{K}(T)} M_{\sigma}$$

— дифференциальный оператор. Композиция $N \circ T$ строится сдедующим образом:

$$N \circ T = N \circ \sum_{\sigma\in\mathcal{K}(T)} M_{\sigma} = \sum_{\sigma\in\mathcal{K}(T)} N \circ M_{\sigma} = f \sum_{\sigma\in\mathcal{K}(T)} \frac{\partial^{|\tau|}}{\partial x^{\tau}} \circ M_{\sigma}.$$

Реализация на *Maple*:

```
diffopComposeWithMonomial := proc( diffMon::table, diffOp::table )::table;
local result::table, diffs::list, f, diffIndex::integer, varIndex::integer;
  if nops( [indices(diffMon)] ) <> 1 then
    error "Incorrect monomial has size %1, indices=%2",
      nops( [indices(diffMon)] ), [indices(diffMon)]
  end if;
  result := copy( diffOp );
  diffs := indices( diffMon )[1];
  f := diffMon[op( diffs )];
  for varIndex from 1 to ivarsCount do
    for diffIndex from 1 to diffs[varIndex] do
      result := diffopDOperator( result, varIndex )
    end do
  end do;
  result := diffopMult( result, f );
  return copy( result );
end proc:
```

Композиция дифференциальных операторов. Имея в распоряжении композицию монома и оператора, мы можем легко построить композицию двух операторов. Пусть даны операторы

$$T_1 = \sum_{\sigma\in\mathcal{K}(T_1)} M_{\sigma}$$

и T_2. Композицию $T_1 \circ T_2$ мы запишем в виде:

$$T_1 \circ T_2 = \left(\sum_{\sigma \in \mathcal{K}(T_1)} M_\sigma \right) \circ T_2 = \sum_{\sigma \in \mathcal{K}(T_1)} \left(M_\sigma \circ T_2 \right).$$

Процедура, реализующая композицию двух операторов, представлена ниже.

```
diffopCompose := proc( diffOp1::table, diffOp2::table )::table;
local result::table, sigma::list, diffMon::table;
  result := table();
  for sigma in indices( diffOp1 ) do
    diffMon := table( [op( sigma ) = diffOp1[ op(sigma) ]] );
    result := diffopPlus( result,
      diffopComposeWithMonomial( diffMon, diffOp2 ) );
  end do;
  diffopRemoveZeroes( result );
  return copy( result );
end proc:
```

Применение дифференциального оператора к уравнению. При реализации мультискобки нам необходимо применять операторы к уравнениям из исходной системы (1). Поэтому мы реализуем следующую процедуру.

Определим действие дифференциального оператора

$$T = \sum_{\sigma \in \mathcal{K}(T)} F_\sigma \frac{\partial^{|\sigma|}}{\partial x^\sigma}$$

на функцию $G(x_1, x_2, \ldots, x_n)$ следующей формулой:

$$T\left(G\right) = \sum_{\sigma \in \mathcal{K}(T)} F_\sigma \frac{\partial^{|\sigma|} G}{\partial x^\sigma}.$$

Исходный код процедуры на *Maple*:

```
diffopOnEquation := proc( diffOp::table, eq )
local result, sigma::list, F, varIndex::integer, i::integer;
  result := 0;
  for sigma in indices( diffOp ) do
    F := eq;
    result := result + diffOp[op( sigma )] * TotDiff( F,
      sigma2vars( sigma ) );
  end do;
  return result;
end proc;
```

1.2. Реализация некоммутативного определителя

Имея в распоряжении построенные вспомогательные процедуры, мы можем реализовать процедуру вычисления некоммутативного определителя в соответствии с определением 2. Мы будем вычислять определитель, раскладывая его по первому столбцу. Ниже мы приводим текст процедуры *Ndet* на языке *Maple*, вычисляющей определитель матрицы, используя рекурсивный алгоритм.

```
Ndet := proc( A::Matrix )::table;
local N::integer, temp::Matrix, result::table, row::integer,
  i::integer, j::integer, tempRow::integer, sign::boolean;
  N := op( 1, A )[1];
  if N <> op( 1, A )[2] then
    error "Matrix in Ndet is not square, %1 <> %2", N, op( 1, A )[2];
  end if;
  if N = 2 then
    return copy( diffopMinus( diffopCompose( A[1, 1], A[2, 2] ),
      diffopCompose( A[2, 1], A[1, 2] ) ) );
  end if;
  temp := Matrix( N - 1, N - 1 );
  result := table();
  sign := true;
  for row from 1 to N do
    sign := not sign;
    tempRow := 0;
    for i from 1 to N do
      if i <> row then
        tempRow := tempRow + 1;
        for j from 2 to N do
          temp[tempRow, j - 1] := A[i, j];
        end do;
      end if;
    end do;
    if sign then
      result := diffopMinus( result, diffopCompose( A[row, 1],
        Ndet( temp ) ) );
    else
      result := diffopPlus( result, diffopCompose( A[row, 1],
        Ndet( temp ) ) );
    end if;
  end do;
  return copy( result );
end proc:
```

Входной параметр функции *Ndet* имеет тип *Matrix*, который определен в *Maple*. Этот тип позволяет хранить матрицу, элементы которой могут иметь любой тип. В качестве результата работы функция возвращает дифференциальный оператор, хранящийся в ассоциативном массиве типа *table.*

1.3. Линеаризация

Важным этапом алгоритма вычисления мультискобки Кругликова–Лычагина является вычисление линеаризации системы

уравнений (1). Чтобы линеаризовать систему дифференциальных уравнений, необходимо линеаризовать каждое из уравнений этой системы. Используя явную формулу из определения (1), реализуем на языке *Maple* линеаризацию одного уравненения.

```
linearizeEquation := proc( F )::Array;
local result::Array, i::integer, sigmas::set, sigma;
  result := Array( 1 .. dvarsCount );
  for i from 1 to dvarsCount do
    result[i] := table();
    sigmas := allSigmas( F, dvars[i] );
    for sigma in sigmas do
      result[i][op( sigma )] := diff( F, dvars[i][op( sigma )] );
    end do;
  end do;
  return result;
end proc:
```

Аргумент функции *linearizeEquation* — это выражение *F*, содержащее независимые переменные (хранятся в глобальном списке *ivars*) и координаты джетов, записанные через зависимые переменные (хранятся в глобальном списке *dvars*) с мультииндексом. Эта функция возвращает объект типа *Array* (определенный в *Maple* тип массива), содержащий дифференциальные операторы.

Размер возвращаемого массива в точности равен числу m — количеству неизвестных функций u_i, которое равно переменной *dvarsCount* — длине списка *dvars*. На *i*-й позиции в этом массиве стоит дифференциальный оператор при *i*-й функции в формуле (3).

Встречающаяся в приведенном тексте функция *allSigmas* — одна из вспомогательных функций, на реализации которых мы не останавливаемся. Данная функция, анализируя выражение, передаваемое ей в первом аргументе *F*, ищет все записанные джетами производные функции, имя которой было передано ей вторым аргументом *dvars*[*i*], и возвращает список всех мультииндексов, стоящих при этом имени в выражении *F*.

Построенную линеаризацию уравнения мы применим для процедуры линеаризация системы уравнений вида (1):

```
linearizeSystem := proc( sys::list )::Matrix;
local result::Matrix, eq::Array, i::integer, j::integer, eqsCount::integer;
  eqsCount := nops( sys );
  result := Matrix( eqsCount, dvarsCount );
  for i from 1 to eqsCount do
    eq := linearizeEquation( sys[i] );
```

```
    for j from 1 to dvarsCount do
      result[i, j] := copy( eq[j] );
    end do;
  end do;
  return result;
end proc;
```

Процедура *linearizeSystem* линеаризует систему уравнений, переданную ей в списке *sys*. Находя линеаризацию каждого уравнения в этом списке, она строит матрицу дифференциальных операторов, имеющую *dvarsCount* столбцов и на единицу большее, чем число уравнений, количество строк.

2. Реализация мультискобки Кругликова–Лычагина

Реализовав все вспомогательные процедуры, мы переходим к реализации определения (3). Ниже представлен исходный текст процедуры *MultiBracket*.

```
MultiBracket := proc( eqSys::list )
local linearization::Matrix, Ndets::Array, result, i::integer;
  linearization := linearizeSystem( eqSys );
  Ndets := allNdets( linearization );
  result := 0;
  for i from 1 to dvarsCount + 1 do
    result := result + (-1)^( (i + 1) mod 2 ) *
      diffopOnEquation( Ndets[i], eqSys[i] );
  end do;
  return simplify( result );
end proc;
```

Входной параметр *eqSys* — список уравнений исходной системы. Процедура возвращает выражение — вычисленную мультискобку. Процедура *allNdets* принимает на вход матрицу дифференциальных операторов размера $m \times (m+1)$, а возвращает список из m некоммутативных определителей, полученных из входной матрицы.

Замечание 1. Оценим сложность вычисления нескольких реализованных функций. Функция *Ndet* вычисляет определитель матрицы разложением по столбцам, следовательно, выполняет $m!$ операций композиции, где m — число неизвестных функций. Функция *MultiBracket* вызывает $m+1$ раз функцию *Ndet*, а затем применяет результат каждого вычисления $m+1$ раз уравнениям исходной системы, т.е. выполняет $(m+1)^2\, m!$ композиций.

Несмотря на то, что формула вычисления мультискобки выполняет значительное количество операций, этот метод получения условий формальной интегрируемости является довольно быстрым, потому что задает *явную* формулу. Сравнение метода скобок с другим методом – дифференциальными базисами Гребнера – приведено в [4].

3. Пример

Рассмотрим следующую задачу: *описать все векторные поля без особых точек на плоскости, первые интегралы которых являются гармоническими функциями.*

Пусть векторные поля имеют вид:

$$X = a(x,y)\frac{\partial}{\partial x} + b(x,y)\frac{\partial}{\partial y}.$$

Задача сводится к исследованию совместности следующей системы дифференциальных уравнения в частных производных:

$$\begin{cases} \Delta u = 0, \\ X(u) = 0, \end{cases}$$

где $\Delta = \partial_{xx} + \partial_{yy}$ — оператор Лапласа. Согласно условию Коши–Римана эта система эквивалентна следующей системе дифференциальных уравнений первого порядка:

$$\begin{cases} u_x - v_y = 0, \\ u_y + v_x = 0, \\ a(x,y)u_x + b(x,y)u_y = 0. \end{cases} \tag{6}$$

В силу того, что векторное поле X не имеет особых точек, можно считать, что функция $a(x,y)$ нигде не обращается в ноль. Тогда без ограничения общности можно положить $a(x,y) \equiv 1$. Система (6) примет вид:

$$\begin{cases} u_x - v_y = 0, \\ u_y + v_x = 0, \\ u_x + b(x,y)u_y = 0. \end{cases} \tag{7}$$

Программа на *Maple*, вычисляющая скобку для системы (7):

```
# подключаем модуль Brackets
with( Brackets ):
# настраиваем внутренние параметры этого модуля:
# списки независимых и зависимых переменных
setup( [x, y], [u, v] ):
# задаем систему уравнений
S := [ u[1, 0] - v[0, 1], u[0, 1] + v[1, 0],
  u[1, 0] + b(x, y) * u[0, 1] ]:
# вычисляем скобку и группируем члены у проивзодных u_x, u_y, u_xy
MlBr := collect( MultiBracket( S ), [ u[0, 1], u[1, 1], u[0, 2] ] );
```

В результате компьютерных вычислений получим мультискобку Кругликова–Лычагина:

$$M = \Delta b u_y + 2b_y u_{yy} + 2b_x u_{xy}, \tag{8}$$

где $\Delta = \partial_{xx} + \partial_{yy}$ — оператор Лапласа. Ограничим скобку M на систему (7), используя третье уравнение $u_x = -b(x,y)u_y$. Дифференцируя его по y, получим $u_{xy} = -b_y u_y - b u_{yy}$. Ограниченную скобку обозначим

$$M_{\mathcal{E}} = (\Delta b - 2b_x b_y)u_y + 2(b_y - b b_x)u_{yy}.$$

Чтобы система (7) была формально интегрируема, необходимо и достаточно, чтобы $M_{\mathcal{E}} \equiv 0$. Видно, что это условие выполнено тогда и только тогда, когда коэффициенты при u_y и u_{yy} равны нулю. Это условие приводит к следующей системе уравнений:

$$\begin{cases} \Delta b = 2b_y b_x, \\ b_y = b b_x. \end{cases} \tag{9}$$

Решая ее, получим

$$b(x,y) = h_1(y)x + h_2(y),$$

где h_1, h_2 — произвольные гладкие функции.

Чтобы исследовать совместность системы (7), перепишем ее в форме, не содержащей функцию $v(x,y)$, и добавим к ней уравнение $M_{\mathcal{E}} = 0$:

$$\begin{cases} u_{xx} + u_{yy} = 0, \\ u_x + b u_y = 0, \\ (\Delta b - 2b_x b_y)u_y + 2(b_y - b b_x)u_{yy} = 0. \end{cases} \tag{10}$$

Продифференцируем второе уравнение этой системы по x и по y и добавим полученные соотношения к системе (10):

$$\begin{cases} u_{xx} + u_{yy} = 0, \\ u_x + bu_y = 0, \\ u_{xx} + b_x u_y + bu_{xy} = 0, \\ u_{xy} + b_y u_y + bu_{yy} = 0, \\ (\Delta b - 2b_x b_y)u_y + 2(b_y - bb_x)u_{yy} = 0. \end{cases}$$

Эта система эквивалентна следующей системе:

$$\begin{cases} u_{xx} + u_{yy} = 0, \\ u_x + bu_y = 0, \\ u_{xy} + b_y u_y + bu_{yy} = 0, \\ -(1 + b^2)u_{yy} + (b_x - bb_y)u_y = 0, \\ 2(b_y - bb_x)u_{yy} + (\Delta b - 2b_x b_y)u_y = 0. \end{cases} \tag{11}$$

Необходимо рассмотреть два случая.

В первом случае, когда $u_y \neq 0$ и $u_{yy} \neq 0$, определитель системы, составленной из четвертого и пятого уравнений системы (11), должен быть равен нулю. Это условие дает следующее соотношение на функцию $b(x, y)$:

$$(b^2 + 1)\Delta b - 2b(b_x^2 + b_y^2) = 0. \tag{12}$$

С учетом соотношения (12) выразим из системы (11) производные u_{xx} и u_{xy} через u_y:

$$u_{xx} = -\frac{(b_x - bb_y)\, u_y}{b^2 + 1},$$

$$u_{xy} = -\left(b_y + \frac{b\,(b_x - bb_y)}{b^2 + 1}\right) u_y.$$

Используя эти соотношения, условие (12) и систему (11), можно показать, что условия формальной интегрируемости системы (11):

$$u_{yyx} = u_{xyy},$$

$$u_{xxy} = u_{xyx}$$

выполняются тождественно.

Рассмотрим второй случай. Пусть $u_y = u_{yy} = 0$. Тогда система (11) сводится к системе:

$$\begin{cases} u_y = 0, \\ u_x = 0, \end{cases}$$

решение которой $u(x, y) = const$. Таким образом мы приходим к следующей теореме.

Теорема 1. *Векторное поле*

$$X = \frac{\partial}{\partial x} + b(x, y)\frac{\partial}{\partial y}$$

обладает не равным константе гармоническим первым интегралом тогда и только тогда, когда функция $b(x, y)$ *удовлетворяет следующему дифференциальному уравнению:*

$$(b^2 + 1)\Delta b - 2b(b_x^2 + b_y^2) = 0.$$

Литература

1. АЛЕКСЕЕВСКИЙ Д.В., ВИНОГРАДОВ А.М., ЛЫЧАГИН В.В. *Основные идеи и понятия дифференциальной геометрии* // Современные проблемы математики. Фундаментальные направления. – 1988. – Т. 28 (Итоги науки и техники ВИНИТИ АН СССР). – С. 5–289.
2. ВИНОГРАДОВ А.М., КРАСИЛЬЩИК И.С., ЛЫЧАГИН В.В. *Введение в геометрию нелинейных дифференциальных уравнений*. – М.: Наука, 1986. – С. 336.
3. КОКС Д., ЛИТТЛ ДЖ., О'ШИ ДЖ. *Идеалы, многообразия и алгоритмы.*. — М.: Мир, 2000. — С. 687.
4. KRUGLIKOV B. *Note on two compatibilty criteria: Jacobi-Mayer bracket vs. differential Groebner basis* // Lobachevskii Journal of Mathematics. – 2006. – Vol. 23. – С. 57–70.
5. KRUGLIKOV B., LYCHAGIN V. *Multi-brackets of differential operators and compatibility of PDE systems* // C. R. Acad. Sci. Paris. – 2006. – Vol. 342. – С. 557–561.
6. *Maple Advanced Programming Guide* / Maplesoft, a division of Waterloo Maple Inc. 1996-2009. – URL: http://www.maplesoft.com/view.aspx?sl=32470 (дата обращения: 12.02.2011).

IMPLEMENTATION OF THE KRUGLIKOV–LYCHAGIN MULTIBRACKET ON MAPLE

Sergey Tychkov, Institute of Control Sciences of RAS, Moscow, postgraduate student, (sergey.lab06@live.ru).

Abstract: We describe implementation of Kruglikov–Lychagin multibracket on Maple. An example is given of application of Kruglikov–Lychagin multibracket for compatibility analysis of a specific system of partial differential equations.

Keywords: differential equations, Kruglikov–Lychagin multibracket, Maple.

Статья представлена к публикации членом редакционной коллегии В. А. Уткиным