Построение базиса дифференциальных инвариантов конечномерной группы Ли средствами компьютерной алгебры

А.А. Талышев
Новосибирский государственный университет
e-mail: tal@academ.org

В работе представлены алгоритм и программа на «Reduce» для построения дифференциальных инвариантов конечномерной группы Ли. Работа программы демонстрируется на группах симметрий уравнений газовой динамики для различных функций состояния.

Содержание

1. Введение

Дифференциальными инвариантами называются инварианты продолженной группы Ли. Для любой группы Ли существует конечный базис дифференциальных инвариантов [1, §24], т.е. такое конечное множество скалярных инвариантов, которое порождает любой дифференциальный инвариант посредством конечного числа функциональных операций и операций инвариантного дифференцирования.

Дифференциальные инварианты позволяют конструировать инвариантные относительно данной группы преобразований системы дифференциальных уравнений и тем

самым производить классификацию таких систем. Дифференциальные инварианты необходимы для построения специального преобразования — группового расслоения дифференциальных уравнений, они также необходимы для построения дифференциально-инвариантных решений.

Программа построения базиса дифференциальных инвариантов реализована на системе «Reduce 3.8» [2] (http://reduce-algebra.sourceforge.net) и включает в себя самодостаточные модули: продолжения векторных полей, вычисления общего ранга системы векторных полей, решения линейных систем обыкновенных дифференциальных уравнений с постоянными коэффициентами, преобразования векторных полей к новым переменным, выделения базиса инвариантов. Для случая линейной зависимости компонент векторных полей от переменных программа может полностью автоматически строить базис.

Работа программы демонстрируется на группах симметрий системы уравнений газовой динамики для различных функций состояния.

2. Продолжение векторного поля

В настоящей работе рассматриваются точечные преобразования, т.е. преобразования пространства зависимых $X = R^n$ и независимых $Y_0 = R^m$ переменных, которые для нужд дифференциальных уравнений продолжаются на пространство полилинейных отображений до соответствующего порядка:

$$Y_i = R^m \otimes S^i R^n, \quad Z_i = X \times Y_0 \times \cdots \times Y_i, \quad i = 0, 1 \ldots, k.$$

Однопараметрическую локальную группу Ли точечных преобразований вполне определяет инфинитезимальный оператор

$$L = \sum_{i=1}^{n} \xi^i(x,u)\partial_{x_i} + \sum_{j=1}^{m} \zeta^j(x,u)\partial_{u^j} \tag{1}$$

продолжение которого на пространство Z_k записывается в виде

$$L^k = L + \sum_{|\alpha|>0}^{|\alpha|\leqslant k} \zeta_\alpha \partial_{u_\alpha}, \tag{2}$$

где

$$\zeta_{\alpha+\gamma_j} = D_j\zeta_\alpha - u_{\alpha+\gamma_i} \sum_{i=1}^{n} D_j\xi^i, \quad j = 1, \ldots, n, \tag{3}$$

$$D_j = \partial_{x_j} + \sum_{|\alpha|\geqslant 0} u_{\alpha+\gamma_j}\partial_{u_\alpha}, \quad j = 1, \ldots, n. \tag{4}$$

Здесь $\alpha = (\alpha^1, \ldots, \alpha^n)$ — целочисленные мультииндексы, $|\alpha| = \sum_{i=1}^n \alpha^i$, γ_j — мультииндекс, у которого j-я компонента равна единице, а остальные равны нулю и $\zeta_{(0,\ldots,0)} = \zeta$.

3. Симметрии уравнений газовой динамики

Уравнения газовой динамики

$$\begin{aligned} u_t + uu_x + vu_y + wu_z + p_x/\rho &= 0, \\ v_t + uv_x + vv_y + wv_z + p_y/\rho &= 0, \\ w_t + uw_x + vw_y + ww_z + p_z/\rho &= 0, \\ \rho_t + u\rho_x + v\rho_y + w\rho_z + \rho(u_x + v_y + w_z) &= 0, \\ p_t + up_x + vp_y + wp_z + A(\rho,p)(u_x + v_y + w_z) &= 0. \end{aligned} \tag{5}$$

при общем виде функции состояния $A(\rho, p)$ допускают 11-мерную алгебру Ли симметрий [1, §11], базис которой обычно записывается в виде:

$$L_1 = \partial_t, \quad L_2 = \partial_x, \quad L_3 = \partial_y, \quad L_4 = \partial_z, \quad L_5 = t\partial_t + x\partial_x + y\partial_y + z\partial_z,$$

$$L_6 = t\partial_x + \partial_u, \quad L_7 = t\partial_y + \partial_v, \quad L_8 = t\partial_z + \partial_w, \quad L_9 = y\partial_x - x\partial_y + v\partial_u - u\partial_v, \tag{6}$$

$$L_{10} = z\partial_x - x\partial_z + w\partial_u - u\partial_w, \quad L_{11} = z\partial_y - y\partial_z + w\partial_v - v\partial_w.$$

При частных видах функции состояния $A(\rho, p)$ возможны расширения алгебры симметрий и дополнительные операторы базиса представлены в следующей таблице:

№	A	операторы	№	A	операторы
1	$pf(p^{1+2\gamma}\rho^{-1})$	$\gamma L'_1 + L'_2$	5	$A_0\rho^\gamma, \quad \gamma \neq 0$	$L'_3, \quad (\gamma-1)L'_1 - 2\gamma L'_2$
2	$f(\rho e^{-2p})$	$L'_1 + L'_3$	6	$\gamma p, \quad \gamma \neq 0$	$L'_1, \quad L'_2$
3	$f(\rho)$	L'_3	7	1	$L'_1, \quad L'_3$
4	$f(p)$	L'_1	8	$5/3p$	$L'_1, \quad L'_2, \quad L'_4$

Здесь A_0 и γ — произвольные константы, f — произвольная функция одной переменной и

$$L'_1 = t\partial_t - u\partial_u - v\partial_v - w\partial_w + 2\rho\partial_\rho, \quad L'_2 = \rho\partial_\rho + p\partial_p, \quad L'_3 = \partial_p,$$

$$L'_4 = t^2\partial_t + tx\partial_x + ty\partial_y + tz\partial_z + (x - tu)\partial_u + (y - tv)\partial_v + (z - tw)\partial_w - 3t\rho\partial_\rho - 5tp\partial_p.$$

При $A \equiv 0$ допускаемая группа будет бесконечномерной, но этот случай здесь не рассматривается.

4. Дифференциальные инварианты

Отображение $\varphi : Z_k \to R$ является инвариантом продолженной группы симметрий G, если и только если $L^k\varphi = 0$ для каждого векторного поля L из алгебры Ли соответствующей группе G. Таким образом для построения всех инвариантов необходимо решить систему уравнений

$$L^k_j\varphi(z_k) = 0, \quad j = 1, \ldots, r, \tag{7}$$

где $L_j, \quad j = 1, \ldots, r$ — базис алгебры симметрий.

Здесь применяется следующий алгоритм решения системы (7). Какое-нибудь векторное поле, например L_1 «распрямляется», т.е. ищется такая замена переменных $z_k \to$

(t'_1, x'_1), в которой поле L_1 переходит в дифференцирование по скалярной переменной t'_1 ($\dim x'_1 = \dim z_k - 1$). Очевидно, что инвариант в новых переменных имеет вид $\varphi'(x'_1)$, т.е. от переменной t'_1 не зависит. Это представление подставляется в остальные уравнения и они расщепляются относительно переменной t'_1. При этом, вообще говоря, новых уравнений будет больше чем $r-1$, но независимых среди них, согласно общей теории [1, §17], будет ровно $r-1$. Только теперь новое семейство векторных полей не обязано образовывать алгебру Ли, но это и не важно. Далее выбирается какое-нибудь векторное поле из новой системы и т.д. После исчерпания всех векторных полей последние переменные x'_r выраженные через исходные переменные z_k и дадут $\dim z_k - r$ независимых скалярных инвариантов. Здесь предполагается, что общий ранг семейства векторных полей $L_1^k, \ldots, L_r^k$ равен r (всегда найдется такое достаточно большое k, при котором это предположение выполняется).

5. Система обыкновенных дифференциальных уравнений с постоянными коэффициентами

Общее решение системы обыкновенных дифференциальных уравнений

$$\frac{dx}{dt} = Ax + b, \quad A \in \mathfrak{L}(R^n, R^n), \quad b \in R^n, \quad x : R \to R^n$$

имеет вид

$$x = e^{At}\bar{x} + c_0 + c_1 t + \cdots + c_k t^k,$$

где k не превышает кратность нулевого собственного значения матрицы A, $\bar{x}$ — произвольный постоянный вектор, а постоянные вектора $c_0, c_1, \ldots, c_k$ определяются из системы уравнений:

$$\begin{aligned} c_1 &= Ac_0 + b, \\ 2c_2 &= Ac_1, \\ &\cdots\cdots\cdots\cdots \\ kc_k &= Ac_{k-1}, \\ 0 &= Ac_k. \end{aligned}$$

Пусть $A = PJP^{-1}$, где J жорданова форма матрицы A, тогда предыдущая система может быть записана в виде:

$$\begin{aligned} \bar{c}_1 &= J\bar{c}_0 + \bar{b}, \\ 2\bar{c}_2 &= J\bar{c}_1, \\ &\cdots\cdots\cdots\cdots \\ k\bar{c}_k &= J\bar{c}_{k-1}, \\ 0 &= J\bar{c}_k. \end{aligned}$$

где $\bar{c}_j = P^{-1}c_j$ и $\bar{b} = P^{-1}b$. Пусть матрица J состоит из клеток Жордана $J_1, \ldots, J_m$, собственные значения и размерности которых равны соответственно $\lambda_1, \ldots, \lambda_m$ и $d_1, \ldots, d_m$. Предыдущая система распадается на независимые подсистемы для каждой клетки Жордана:

$$\begin{aligned} \bar{c}_{l1} &= J_l\bar{c}_{l0} + \bar{b}_l, \\ 2\bar{c}_{l2} &= J_l\bar{c}_{l1}, \\ &\cdots\cdots\cdots\cdots \\ k\bar{c}_{lk} &= J_l\bar{c}_{lk-1}, \\ 0 &= J_l\bar{c}_{lk} \end{aligned} \quad , \quad l = 1, \ldots, m.$$

Если $\lambda_l \neq 0$, то подсистема имеет единственное решение: $\bar{c}_{l0} = -J_l^{-1}\bar{b}_l$ и $\bar{c}_{lj} = 0$ для $j > 0$. Если $\lambda_l = 0$, то имеется произвол в построении решения подсистемы и одним из решений будет (а больше одного и не требуется): $\bar{c}_{l0} = 0$ и $\bar{c}_{lj} = J_l^{j-1}\bar{b}_l/j!$ для $j > 0$.

6. Описание компонент программы

6.1. Продолжение алгебры

Функция compare_lists$(s1, s2, n)$ возвращает 1, если совпадают первые n элементов списков $s1$ и $s2$. В противном случае функция возвращает 0.

```
procedure compare_lists(s1, s2, n);
begin
  scalar z, j;
  z := 1;
  for j := 1:n do
  begin
    if part(s1,j) neq part(s2,j) then
    begin
      z := 0;
      goto m1;
    end;
  end;
  m1:;
  return z;
end;
```

Функция copy_lists$(s1, n)$ возвращает список составленный из первых n элементов списка $s1$.

```
procedure copy_lists(s1, n);
begin
  scalar s2, j;
  s2 := {};
  for j := 1:n do
  s2 := append(s2,{part(s1,j)});
  return s2;
end;
```

Функция do_ind(k, n) создает список всех мультииндексов длины n до k-го порядка включительно. Список содержит не только мультииндексы, но и информацию о том, какой операцией и из какого мультииндекса предыдущего порядка данный мультииндекс может быть получен. Для каждого мультииндекса $\alpha = (\alpha_1, \ldots, \alpha_n)$ порядка i существуют мультииндекс β порядка $i-1$ и целое число j из интервала $[1, n]$ такие, что $\alpha = \beta + \gamma_j$ (γ_j — мультииндекс, у которого j-я компонента равна единице, а остальные равны нулю). Мультииндекс β и число j определяются неоднозначно, но для реализации формулы продолжения (3) необходим какой-нибудь один вариант. Например, вызов функции do_ind$(2, 2)$; возвратит список: {{{0,0,0,0}}, {{1,0,1,1},{0,1,1,2}},

{{2,0,1,1},{1,1,1,2},{0,2,2,2}}}. Здесь первых два элемента списков третьего уровня — мультииндексы, третий элемент — номер мультииндекса в списке мультииндексов предыдущего порядка и четвертый элемент — число j.

```
procedure do_ind(k, n);
begin
  scalar s1, s2, s3, v, bb, j, k1, i1, i2;
  s1 := {};
  for j := 1:(n+2) do
  s1 := append(s1, {0});
  v := {{s1}};
  for k1 := 1:k do
  begin
    s1 := {};
    for j := 1:length(part(v,k1)) do
    begin
      s2 := copy_lists(part(v,k1,j), n);
      for i2 := 1:n do
      begin
        s3 := part(s2,i2) := part(s2,i2) + 1;
        bb := 1;
        for i1 := 1:length(s1) do
        begin
          if compare_lists(part(s1,i1), s3, n) = 1 then
          begin
            bb := 0;
            goto m1;
          end;
        end;
        m1:;
        if bb = 1 then
        begin
          s3 := append(s3,{j,i2});
          s1 := append(s1,{s3});
        end;
      end;
    end;
    v := append(v,{s1});
  end;
  return v;
end;
```

Функция do_var$(u, indep, s)$ создает переменную из продолженного пространства Z_k, используя зависимую переменную u, список независимых переменных $indep$ и мультииндекс s. Например, вызов функции do_var$(u, \{x, y\}, \{2, 1\})$; вернет переменную $uxxy$.

```
procedure do_var(u, indep, s);
begin
```

```
  scalar w, j, n, j1;
  n := length(indep);
  w := u;
  for j := 1:n do
  begin
    for j1 := 1:part(s,j) do
    w := mkid(w,part(indep,j));
  end;
  return w;
end;
```

Функция prol_vars($indep, dep, l\_ind$) возвращает список переменных пространства Z_k, где $indep$ — список независимых переменных, dep — список зависимых переменных и l_ind — список мультииндексов. Например, вызов функции prol_vars($\{x, y\}$, $\{u, v\}$, do_ind$(2, 2)$); вернет список:

$$\{x, y, u, v, ux, vx, uy, vy, uxx, vxx, uxy, vxy, uyy, vyy\}.$$

```
procedure prol_vars(indep, dep, l_ind);
begin
  scalar s_out, n, m, s, s1, u, j, j1, j2, j3;
  s_out := indep;
  n := length(indep);
  m := length(dep);
  for j1 := 1:length(l_ind) do
  begin
    s := part(l_ind, j1);
    for j2 := 1:length(s) do
    begin
      for j := 1:m do
      begin
        u := do_var(part(dep,j), indep, part(s, j2));
        s_out := append(s_out, {u});
      end;
    end;
  end;
  return s_out;
end;
```

Функция full_dif($indep, dep, l\_ind$) возвращает список списков компонент векторов полного дифференцирования (4) (D_j, $j = 1, \ldots, n$), где $indep$ — список независимых переменных, dep — список зависимых переменных и l_ind — список мультииндексов. Например, команда full_dif($\{x, y\}$, $\{u, v\}$, do_ind$(1, 2)$); вернет список:

$$\{\{1, 0, ux, vx, uxx, vxx, uxy, vxy\}, \{0, 1, uy, vy, uxy, vxy, uyy, vyy\}\}.$$

```
procedure full_dif(indep, dep, l_ind);
begin
```

```
  scalar s_out, n, m, s, s1, s2, s3, u, j, j0, j1, j2;
  s_out := {};
  n := length(indep);
  m := length(dep);
  for j0 := 1:n do
  begin
    s3 := {};
    for j := 1:n do
    if j = j0 then
    s3 := append(s3, {1})
    else
    s3 := append(s3, {0});
    for j1 := 1:length(l_ind) do
    begin
      s := part(l_ind, j1);
      for j2 := 1:length(s) do
      begin
        for j := 1:m do
        begin
          s1 := copy_lists(part(s,j2), n);
          s2 := part(s1,j0) := part(s1,j0) + 1;
          u  := do_var(part(dep,j), indep, s2);
          s3 := append(s3, {u});
        end;
      end;
    end;
    s_out := append(s_out, {s3});
  end;
  return s_out;
end;
```

Функция field_act$(s1, s2, u)$ возвращает результат действия векторного поля на выражение u, где $s1$ — список переменных и $s2$ — список компонент поля.

```
procedure field_act(s1, s2, u);
begin
  scalar s_out, j;
  s_out := 0;
  for j := 1:length(s1) do
  begin
    if part(s2, j) neq 0 then
    begin
      s_out := s_out + part(s2,j)*df(u, part(s1,j));
    end;
  end;
  return s_out;
end;
```

Функция commut$(v, s1, s2)$ возвращает список компонент коммутатора двух векторных полей, где v — список переменных, $s1$ — список компонент первого поля и $s2$ — список компонент второго поля.

```
procedure commut(v, s1, s2);
begin
  scalar s_out, j, u;
  s_out := {};
  for j := 1:length(v) do
  begin
    u := field_act(v, s1, part(s2,j)) - field_act(v, s2, part(s1,j));
    s_out := append(s_out, {u});
  end;
  return s_out;
end;
```

Функция times_field(s, a) возвращает список компонент векторного поля s умноженных на выражение a.

```
procedure times_field(s, a);
begin
  scalar s_out, j;
  s_out := {};
  for j := 1:length(s) do
  s_out := append(s_out, {a*part(s,j)});
  return s_out;
end;
```

Функция plus_field$(s1, s2, a1, a2)$ возвращает линейную комбинацию двух векторных полей, где $s1$ — список компонент первого поля, $s2$ — список компонент второго поля, $a1$ — первый множитель и $a2$ — второй множитель.

```
procedure plus_field(s1,s2,a1,a2);
begin
  scalar s_out, j;
  s_out := {};
  for j := 1:length(s1) do
  s_out := append(s_out, {a1*part(s1,j)+a2*part(s2,j)});
  return s_out;
end;
```

Функция prol_al$(s_in, indep, dep, l_ind, vars, full_d)$, применяя формулы (3) возвращает список компонент продолженного векторного поля (2), где s_in — компоненты исходного векторного поля L, $indep$ — список независимых переменных, dep — список зависимых переменных, l_ind — список мультииндексов, $vars$ — список переменных продолженного пространства Z_k, $full\_d$ — список компонент операторов полного дифференцирования.

```
procedure prol_al(s_in, indep, dep, l_ind, vars, full_d);
begin
  scalar s_out, n, m, m1, m2, m21, m3, m4, s, s0, s2, s3,
         j, j1, j2, j3, u, u1, u2;
  s_out := s_in;
  n := length(indep);
  m := length(dep);
  m1 := n;
  m21 := m;
  for j1 := 2:length(l_ind) do
  begin
    m2 := 0;
    s := part(l_ind, j1);
    for j2 := 1:length(s) do
    begin
      m3 := part(s,j2,n+1);
      m4 := part(s,j2,n+2);
      for j := 1:m do
      begin
        u1 := part(s_out, m1 + (m3-1)*m + j);
        s0 := part(full_d,m4);
        u := field_act(vars, s0, u1);
        s2 := copy_lists(part(l_ind,j1-1,m3), n);
        for j3 := 1:n do
        begin
          s3 := part(s2,j3) := part(s2,j3) + 1;
          u2 := do_var(part(dep,j), indep, s3);
          u := u - u2*field_act(vars, s0, part(s_in, j3));
        end;
        m2 := m2 + 1;
        s_out := append(s_out, {u});
      end;
    end;
    m1 := m1 + m21;
    m21 := m2;
  end;
  return s_out;
end;
```

6.2. Решение системы обыкновенных дифференциальных уравнений

Функция split_ode(var, gr) строит систему обыкновенных дифференциальных уравнения в нормальной форме для векторного поля, где var — список переменных, а gr — список компонент поля. Одновременно функция выявляет случаи тривиального расщепления системы. Результат выдает в виде двухуровнего списка. Каждый список второго уровня представляет независимую подсистему уравнений: первый элемент — количество уравнений, далее каждый четный элемент — переменная из левой части системы,

а нечетный — правая часть этого уравнения. Например, команда

$$\text{split\_ode}(\{x, y, z, u, v, w\}, \{-y, x, 0, -v, u, 0\});$$

вернет список:

$$\{\{2, x, -y, y, x\}, \{2, u, -v, v, u\}, \{1, z, 0\}, \{1, w, 0\}\}.$$

Данный пример реализует применение функции к преобразованию вращения в шестимерном пространстве с векторным полем

$$-y\partial_x + x\partial_y - v\partial_u + u\partial_v.$$

Соответствующая система дифференциальных уравнений здесь будет следующая:

$$\begin{cases} \dot{x} = -y, \\ \dot{y} = x, \end{cases} \quad \begin{cases} \dot{u} = -v, \\ \dot{v} = u, \end{cases} \quad \dot{z} = 0, \quad \dot{w} = 0.$$

```
procedure split_ode(var, gr);
begin
  scalar s, ind, j_cur, s_out, j1, j2;
  s_out := {};
  ind := {};
  for j:=1:length(gr) do
  ind := append(ind, {0});
  for j := 1:length(gr) do
  if part(gr,j) neq 0 and part(ind,j) = 0 then
  begin
    s := {0};
    j_cur :={j};
    while length(j_cur) > 0 do
    begin
      j2 := first(j_cur);
      ind := part(ind,j2) := 2;
      for j1 := 1:length(gr) do
      if df(part(gr,j2),part(var,j1)) neq 0 and part(ind,j1) = 0 then
      begin
        j_cur := append(j_cur,{j1});
        ind := part(ind,j1) := 2;
      end;
      s := append(s, {part(var,j2)});
      s := append(s, {part(gr,j2)});
      s := part(s,1) := part(s,1) + 1;
      ind := part(ind,j2) := 1;
      j_cur := rest(j_cur);
    end;
    s_out := append(s_out, {s});
  end;
  for j:=1:length(gr) do
  if part(ind,j) = 0 then
```

```
  s_out := append(s_out, {{1,part(var,j),0}});
  return s_out;
end;
```

Функция get_jcell($jmat$) возвращает список с исчерпывающей информацией о клетках Жордана жордановой матрицы $jmat$. Список имеет вид: $\{\{d_1, \lambda_1\}, \dots\}$, где d_j — размерность, а λ_j — собственное число j-ой клетки Жордана.

```
procedure get_jcell(jmat);
begin
  scalar s_out, s, n, j, j1;
  s_out := {};
  mat2 := jmat;
  n := part(length(mat2),1);
  j1 := 0;
  for j := 1:n do
  begin
    if j = n or mat2(j,j+1) neq 1 then
    begin
      s := {j1+1,mat2(j,j)};
      s_out := append(s_out, {s});
      j1 := 0;
    end
    else
    j1 := j1 + 1;
  end;
  return s_out;
end;
```

Функция do_ident_mat(n) возвращает единичную матрицу размерности n.

```
procedure do_ident_mat(n);
begin
  scalar j1, j2;
  matrix mat_out(n,n);
  for j1:=1:n do
  for j2:=1:n do
  if j1 = j2 then
  mat_out(j1,j2):=1
  else
  mat_out(j1,j2):=0;
  return mat_out;
end;
```

Функция do_jordc_mat(n, lam) возвращает клетку Жордана размерности n с собственным числом lam.

```
procedure do_jordc_mat(n, lam);
begin
  scalar j1;
  mat_out := do_ident_mat(n);
  mat_out := lam*mat_out;
  for j1 := 2:n do
  mat_out(j1-1,j1) := 1;
  return mat_out;
end;
```

Функция solv_j(equ, tt) возвращает общее решение системы обыкновенных дифференциальных уравнений equ, используя формулы раздела 6.2. Аргумент tt используется в качестве независимой переменной системы обыкновенных дифференциальных уравнений. Первый элемент списка equ — количество уравнений, далее каждый четный элемент — переменная из левой части системы, а нечетный — правая часть этого уравнения. Функция генерирует идентификаторы произвольных постоянных общего решения, добавляя к каждой зависимой переменой символ «1». Перед вызовом функции должен быть инициирован список inf_jordan — в него функция помещает информацию о клетках Жордана системы. Для приведения матрицы к форме Жордана используется функция «jordan» из пакета «normform», поэтому перед описанием функции загружается этот пакет.

Например, команда solv_j($\{2, x, -y, y, x\}, tt$); вернет решение $\{x = cos(tt) * x1 - \sin(tt) * y1, y = \cos(tt) * y1 + sin(tt) * x1\}$.

```
load_package normform;
procedure solv_j(equ, tt);
begin
  scalar s_out, j1, j2, j3, j4, n, lam, u1, u2, s, s1, ind0;
  n := part(equ,1);
  inf_j :={};
  matrix amat(n,n), bmat(n,1), bmat2(n,1);
  for j1 := 1:n do
  begin
    u1 := part(equ,2*j1+1);
    for j2 := 1:n do
    begin
      u2 := part(equ,2*j2);
      amat(j1,j2) := df(u1,u2);
      u1 := u1 - amat(j1,j2)*u2;
    end;
    bmat(j1,1) := u1;
    bmat2(j1,1) := mkid(part(equ,2*j1),1);
  end;
  s := jordan(amat);
  amat := part(s,1);
  s1 := get_jcell(s);
  inf_jordan := append(inf_jordan, {s1});
```

```
bmat1 := part(s,3)*bmat;
bmat2 := part(s,3)*bmat2;
j4 := 0;
for j1:=1:length(s1) do
begin
  j3  := part(s1,j1,1);
  lam := part(s1,j1,2);
  if j3 > 1 then
  begin
    matrix amat3(j3,j3), bmat3(j3,1);
    amat3 := do_jordc_mat(j3, 0);
    amat4 := do_ident_mat(j3);
    amat5 := amat4;
    for j2 := 1:(j3 - 1) do
    begin
      amat4 := amat4*amat3*tt/j2;
      amat5 := amat5 + amat4;
    end;
    for j2 := 1:j3 do
    bmat3(j2,1) := bmat2(j4 + j2,1);
    bmat3 := amat5*bmat3;
    if lam neq 0 then
    begin
      bmat3 := exp(repart(lam)*tt)*(cos(impart(lam)*tt)
             + i*sin(impart(lam)*tt))*bmat3;
    end;
    matrix bmat4(j3,1);
    ind0 := 1;
    for j2 := 1:j3 do
    begin
      bmat4(j3,1) := bmat1(j4 + j2,1);
      if bmat4(j3,1) neq 0 then
      ind0 := 0;
    end;
    if ind0 = 1 then
    begin
      amat3 := do_jordc_mat(j3, lam);
      if lam = 0 then
      begin
        bmat4 := bmat4*tt;
        bmat3 := bmat3 + bmat4;
        for j2 := 2:j3 do
        begin
          bmat4 := amat3*bmat4*tt/j2;
          bmat3 := bmat3 + bmat4;
        end;
      end
```

```
        else
        bmat3 := bmat3 - amat3^(-1)*bmat4;
      end;
      for j2 := 1:j3 do
      bmat2(j4 + j2,1) := bmat3(j2,1);
    end
    else
    begin
      j2 := j4 + 1;
      if lam = 0 then
      bmat2(j2,1) := bmat2(j2,1) + bmat1(j2,1)*tt
      else
      begin
        bmat2(j2,1) := exp(repart(lam)*tt)*(cos(impart(lam)*tt)
                    + i*sin(impart(lam)*tt))*bmat2(j2,1) + bmat1(j2,1)/lam;
      end;
    end;
    j4 := j4 + j3;
  end;
  bmat2 := part(s,2)*bmat2;
  s_out := {};
  for j1 := 1:n do
  s_out := append(s_out, {part(equ,2*j1) = bmat2(j1,1)});
  return s_out;
end;
```

Функция solve_ode(equ, tt) возвращает общее решение системы обыкновенных дифференциальных уравнений *equ*, состоящей из совокупности независимых подсистем. Скалярные подсистемы функция решает сама, а для решения векторных подсистем обращается к функции solv_j. Аргумент *equ* является списком подсистем, каждый элемент которого Аргумент *tt* используется в качестве независимой переменной системы обыкновенных дифференциальных уравнений. Функция генерирует идентификаторы произвольных постоянных общего решения, добавляя к каждой зависимой переменой символ «1». Перед вызовом функции должен быть инициирован список inf_jordan — в него функция помещает информацию о клетках Жордана системы.

```
procedure solve_ode(equ, tt);
begin
  scalar s, a, a1, x1, s_out;
  list_out := {};
  inf_jordan := {};
  for j:=1:length(equ) do
  begin
    if part(equ,j,1) = 1 then
    begin
      inf_jordan := append(inf_jordan, {{{0,0}}});
      x1 := mkid(part(equ,j,2), 1);
```

```
      if part(equ,j,3) = 0 then
      begin
        s_out := append(s_out, {part(equ,j,2) = x1});
      end
      else
      begin
        a := df(part(equ,j,3), part(equ,j,2));
        if a = 0 then
        begin
          s_out := append(s_out,{part(equ,j,2) = part(equ,j,3)*tt+x1});
        end
        else
        begin
          a1 := part(equ,j,3) - a*part(equ,j,2);
          s_out := append(s_out, {part(equ,j,2) = x1*exp(a*tt)-a1/a});
        end;
      end;
    end
    else
    begin
      s := solv_j(part(equ,j), tt);
      s_out := append(s_out, s);
    end;
  end;
  return s_out;
end;
```

Функция get_out_var(equ, sol, tt) находит такую константу из представления общего решения sol системы уравнений equ, чтобы фиксация этой константы на определенном значении позволяла разрешить общее решение относительно остальных констант и переменной tt.

```
procedure get_out_var(equ, sol, tt);
begin
  scalar s_out, vars1, s1, s2, j, j1, j2, j3, j4, j5, a, b, u, u1;
  s_out := {0};
  j1 := 1;
  for j := 1:length(equ) do
  begin
    if part(equ,j,1) = 1 then
    begin
      a := df(rhs(part(sol,j1)), tt);
      if a neq 0 then
      begin
        u := lhs(part(sol,j1));
        u1 := mkid(u,1);
        if df(a,tt) = 0 then
```

```
      s_out := {1,j,{u1 = 0},{tt = u/a}}
      else
      begin
        %b := df(rhs(part(sol,j1)), part(vars1,j1));
        b := df(part(equ,j,3), part(equ,j,2));
        s_out := {2,j,{u1 = 1}, {exp(tt) = (u
               - (rhs(part(sol,j1)) - u1*exp(b*tt)))^(1/b)}};
      end;
      goto m1;
    end;
  end;
  j1 := j1 + part(equ,j,1);
end;
j1 := 1;
for j := 1:length(equ) do
begin
  if part(equ,j,1) = 2 then
  begin
    vars1 := {mkid(lhs(part(sol,j1)),1),mkid(lhs(part(sol,j1 + 1)),1)};
    s := part(inf_jordan, j);
    if part(s, 1, 1) = 2 and part(s, 1, 2) = 0 then
    begin
      for j2 := 1:2 do
      for j3 := 1:2 do
      begin
        u := sub({part(vars1,j3) = 0}, rhs(part(sol,j2)));
        if df(u,tt) neq 0 and df(u,tt,2) = 0 then
        begin
          s2 := solve({u=lhs(part(sol,j2))},tt);
          j4 := (j2 + 1) mod 2 + 1;
          j5 := (j3 + 1) mod 2 + 1;
          u := sub({part(vars1,j3) = 0}, rhs(part(sol,j4)));
          u := sub(s2,u);
          s3 := solve({u=lhs(part(sol,j4))},part(vars1,j5));
          s_out := {3,j,{part(vars1,j3) = 0,part(s3,1)}};
          s_out := append(s_out, {s2});
          goto m1;
        end;
      end;
    end
    else
    if part(s, 1, 1) = 1 and abs(impart(part(s, 1, 2))) = 1
       and repart(part(s, 1, 2)) = 0 then
    begin
      s1 := {part(vars1,1) = 0};
      s2 := solve({lhs(part(sol,j1))=sub(s1, rhs(part(sol,j1))),
            lhs(part(sol,j1+1))=sub(s1, rhs(part(sol,j1+1)))},
```

```
              {cos(tt),sin(tt)});
        s1 := {part(vars1,2) = sqrt((lhs(part(sol,j1)))^2
            + (lhs(part(sol,j1+1)))^2)};
        s_out := {4,j,{part(vars1, 1) = 0,part(s1,1)}};
        s_out := append(s_out, sub(s1,s2));
        goto m1;
      end;
    end;
    j1 := j1 + part(equ,j,1);
  end;
  m1:;
  return s_out;
end;
```

Функция inverse_sol(equ, sol, o_var, tt) фиксирует выбранную функцией get_out_var константу и разрешает общее представление решения sol системы equ относительно остальных констант и переменной tt. Через аргумент o_var передаются результаты работы функции get_out_var.

```
procedure inverse_sol(equ, sol, o_var, tt);
begin
  scalar s_out, vars1, vars2, s, j, j1, j2, n, u, u1;
  if part(o_var,1) < 3 then
  s_out := {}
  else
  s_out := {part(o_var,3,2)};
  i1 := 0;
  for j := 1:length(equ) do
  begin
    n := part(equ,j,1);
    if j neq part(o_var, 2) then
    begin
      matrix amat(n,n), bmat(n,1);
      vars1 := {};
      vars2 := {};
      for j1 := 1:n do
      begin
        u := lhs(part(sol,i1 + j1));
        u1 := mkid(u,1);
        vars1 := append(vars1, {u});
        vars2 := append(vars2, {u1});
      end;
      for j1 := 1:n do
      begin
        u := rhs(part(sol,i1 + j1));
        bmat(j1,1) := lhs(part(sol,i1 + j1)) - u;
        for j2 := 1:n do
```

```
        begin
          amat(j1,j2) := df(u,part(vars2,j2));
          bmat(j1,1)  := bmat(j1,1) + amat(j1,j2)*part(vars2,j2);
        end;
      end;
      amat := sub({tt = -tt},amat);
      bmat := amat*bmat;
      s := {};
      for j1 := 1:n do
      s := append(s, {part(vars2,j1) = bmat(j1,1)});
      s_out := append(s_out, s);
    end;
    i1 := i1 + n;
  end;
  return s_out;
end;
```

Функция get_impart_l($s0$) возвращает список абсолютных значений мнимых частей собственных значений матрицы системы уравнений.

```
procedure get_impart_l(s0);
begin
  scalar s_out, j, j1, j2, u, b;
  s_out := {};
  for j := 1:length(s0) do
  begin
    for j1 := 1:length(part(s0,j)) do
    begin
      u := impart(part(s0,j,j1,2));
      if u neq 0 then
      begin
        b := 1;
        for j2 := 1:length(s_out) do
        if abs(u) = part(s_out,j2) then
        begin
          b := 0;
          goto m1;
        end;
        m1:;
        if b = 1 then
        s_out := append(s_out, {u});
      end;
    end;
  end;
  return s_out;
end;
```

Функция n_cos(n, tt) возвращает выражение $\cos(n\,tt)$ через $\cos(tt)$ и $\sin(tt)$.

```
procedure n_cos(n, tt);
begin
  scalar u;
  if n = 1 then
  u := cos(tt)
  else
  u := n_cos(n - 1, tt)*cos(tt) - n_sin(n - 1, tt)*sin(tt);
  return u;
end;
```

Функция n_sin(n, tt) возвращает выражение $\sin(n\,tt)$ через $\cos(tt)$ и $\sin(tt)$.

```
procedure n_sin(n, tt);
begin
  scalar u;
  if n = 1 then
  u := sin(tt)
  else
  u := n_cos(n - 1, tt)*sin(tt) + n_sin(n - 1, tt)*cos(tt);
  return u;
end;
```

Функция test_zero(s) возвращает 1, если все элементы списка s равны нулю.

```
procedure test_zero(s);
begin
  scalar j, b_out;
  b_out := 1;
  for j := 1:length(s) do
  if part(s,j) neq 0 then
  begin
    b_out := 0;
    goto m1;
  end;
  m1:;
  return b_out;
end;
```

Функция test_linear$(vars, s)$ возвращает 1, если все элементы списка s линейны по переменным из списка $vars$.

```
procedure test_linear(vars, s);
begin
  scalar j1, j2, j3, b_out;
  b_out := 1;
  for j1 := 1:length(s) do
  begin
    for j2 := 1:length(vars) do
```

```
    for j3 := 1:length(vars) do
    if df(part(s,j1),part(vars,j2),part(vars,j3)) neq 0 then
    begin
      b_out := 0;
      goto m1;
    end;
  end;
  m1:;
  return b_out;
end;
```

Функция test_depend(s, tt) возвращает 1, если все элементы списка s не зависят от переменной tt.

```
procedure test_depend(s, tt);
begin
  scalar j1, b_out;
  b_out := 0;
  for j1 := 1:length(s) do
  begin
    if df(part(s,j1),tt) neq 0 then
    begin
      b_out := 1;
      goto m1;
    end;
  end;
  m1:;
  return b_out;
end;
```

Если список gr содержит единственный ненулевой элемент и он является константой по отношению к переменным из списка var, то функция test_1(var, gr) возвращает номер этого элемента. В противном случае функция возвращает ноль.

```
procedure test_1(var, gr);
begin
  scalar j_out, j, j1;
  j_out := 0;
  for j := 1:length(var) do
  if part(gr,j) neq 0 then
  begin
    if j_out > 0 then
    begin
      j_out := 0;
      goto m1;
    end;
    for j1 := 1:length(var) do
    if df(part(gr,j), part(var,j1)) neq 0 then
```

```
    goto m1;
    j_out := j;
  end;
  m1:;
  return j_out;
end;
```

Функция test_line_dep(s, tt) предназначена для расщепления элементов списка s по переменной tt в случае их линейной зависимости по tt.

```
procedure test_line_dep(s, tt);
begin
  scalar s_out, s1, s2, s3, j, j1;
  s_out := {};
  s1 := sub({tt = 0}, s);
  if test_zero(s1) = 0 then
  s_out := append(s_out,{s1});
  s2:={};
  for j1 := 1: length(s) do
  s2:=append(s2,{df(part(s,j1),tt)});
  if test_zero(s2) = 0 then
  begin
    s3 := sub({tt = 0}, s2);
    if test_zero(s3) = 0 then
    s_out := append(s_out,{s3});
  end;
  return s_out;
end;
```

Функция test_1_out$(vars, gr, j0, j1)$ исключает векторное поле с номером $j0$ из списка полей gr и переменную с номером $j1$ из списка переменных $vars$ и проецирует остальные векторные поля на новый набор переменных.

```
procedure test_1_out(var, gr, j0, j1);
begin
  scalar s_out, s1, s2, s3, j, j2, tt;
  tt := part(var,j1);
  s_out := {};
  for j := 1: length(gr) do
  if j neq j0 then
  begin
    s1 := {};
    for j2 := 1: length(var) do
    if j2 neq j1 then
    s1 := append(s1, {part(gr,j,j2)});
    s2 := sub({tt = 0}, s1);
    if test_zero(s2) = 0 then
    s_out := append(s_out,{s2});
```

```
    s2:={};
    for j2 := 1: length(s1) do
    s2:=append(s2,{df(part(s1,j2),tt)});
    if test_zero(s2) = 0 then
    begin
      s3 := sub({tt = 0}, s2);
      if test_zero(s3) = 0 then
      s_out := append(s_out,{s3});
    end;
  end;
  return s_out;
end;
```

Функция test_1_all($vars, gr$) находит все векторные поля в списке полей gr, которые не требуется «распрямлять» (например, поля соответствующие преобразованиям переноса) и проецирует остальные векторные поля и переменные $vars$ на соответствующее подпространство.

```
procedure test_1_all(vars, gr);
begin
  scalar bb, vars1, vars2, gr1, gr2, j, j1, j2;
  gr1 := gr;
  vars1 := vars;
  bb := 1;
  while bb > 0 do
  begin
    bb := 0;
    for j := 1:length(gr1) do
    begin
      j1 := test_1(vars1, part(gr1,j));
      if j1 > 0 then
      begin
        bb := 1;
        gr2 := test_1_out(vars1, gr1, j, j1);
        gr1 := gr2;
        vars2 := {};
        for j2 := 1:length(vars1) do
        if j2 neq j1 then
        vars2 := append(vars2, {part(vars1,j2)});
        vars1 := vars2;
        goto m1;
      end;
    end;
    m1:;
  end;
  return {vars1, gr1};
end;
```

```
pause;
ss := test_0_all(vars, gr1)$
vars1:=part(ss,1);
gr:=part(ss,2);

%end;


gr1 := gr;
%end;
```

6.3. Построение полиномиальных инвариантов

Данный раздел содержит описание набора процедур предназначенных для построения полиномиальных инвариантов множества векторных полей с полиномиальными коэффициентами. Используется вполне естественный алгоритм: конструируется однородный полином с неопределенными коэффициентами, результат действия векторных полей на этот полином приравнивается нулю. Это приводит к линейной системе уравнений на коэффициенты полинома. Находится полная система линейно-независимых решений. Подстановка этих решений в полином дает набор инвариантов.

Служебная функция next_ind(s, n) возвращает следующий список в множестве лексикографически упорядоченных списков из набора элементов $1, 2, \ldots, n$. Например, команда next_ind$(\{1, 2, 2\}, 2)$; вернет список {2,1,1}.

```
procedure next_ind(s, n);
begin
  scalar s_out, j, j1;
  j := length(s);
  while (j > 0) and (part(s, j) = n)  do
  j := j - 1;
  if j > 0 then
  begin
    j1 := part(s,j) + 1;
    s_out := s;
    while j <= length(s) do
    begin
      s_out := part(s_out,j) := j1;
      j := j + 1;
    end;
  end
  else
  s_out := {};
  return s_out;
end;
```

Функция func_pol(a, v, n) генерирует однородный полином общего вида степени n от переменных из списка v и возвращает двухэлементный список. Первый элемент

этого списка — искомый полином, а второй элемент — список коэффициентов данного полинома. Параметр a используется в качестве первого символа при конструировании коэффициентов. Например, команда func_pol$(a, \{x, y, z\}, 2)$; вернет список $\{a\_1\_1 * x * *2 + a\_1\_2 * x * y + a\_1\_3 * x * z + a\_2\_2 * y * *2 + a\_2\_3 * y * z + a\_3\_3 * z * *2, \{a\_1\_1, a\_1\_2, a\_1\_3, a\_2\_2, a\_2\_3, a\_3\_3\}\}$

```
procedure func_pol(a, v, n);
begin
  scalar s_out, ind, v1, j, u, u1, nv;
  ind := {};
  for j:=1:n do
  ind := append(ind, {1});
  v1 := {};
  u  := 0;
  nv := length(v);
  while length(ind) > 0 do
  begin
    a1 := a;
    u1 := 1;
    for j:=1:n do
    begin
      a1 := mkid(mkid(a1,_),part(ind,j));
      u1 := u1*part(v,part(ind,j));
    end;
    u  := u + a1*u1;
    v1 := append(v1,{a1});
    ind := next_ind(ind, nv);
  end;
  s_out := {u,v1};
  return s_out;
end;
```

Процедура all_coeff(ur, var, j) находит все коэффициенты полиномов из списка ur от переменных из списка var и помещает их в глобальный список $list\_out$. Параметр j играет роль счетчика переменных при рекурсивном вызове процедуры.

```
procedure all_coeff(ur, var, j);
begin
  scalar ur1, v, i1, i2, i3;
  for i1:=1:length(ur) do
  begin
    ur1:=coeff(part(ur, i1), part(var, j));
    if j < length(var) then
    begin
      all_coeff(ur1, var, j + 1)
    end
    else
    begin
```

```
      for i2:=1:length(ur1) do
      begin
        v := part(ur1, i2);
        if v neq 0 then
        begin
          b := 1;
          for i3:=1:length(list_out) do
          begin
            if v = part(list_out,i3) then
            begin
              b := 0;
              goto m1;
            end;
          end;
          m1:;
          if b = 1 then
          list_out := append(list_out, {v});
        end;
      end;
    end;
  end;
end;
```

Функция get_eqw($vars, gr, u$) действует векторными полями из списка gr в пространстве переменных из списка $vars$ на полином u, выделяет все коэффициенты полученных полиномов, приравнивает их нулю и возвращает список этих уравнений.

```
procedure get_eqw(vars, gr, u);
begin
  scalar s_out, j, v;
  list_out := {};
  for j := 1:length(gr) do
  begin
    v := field_act(vars,part(gr,j),u);
    all_coeff({v},vars,1);
  end;
  s_out := {};
  for j:=1:length(list_out) do
  s_out := append(s_out, {part(list_out,j) = 0});
  return s_out;
end;
```

Функция get_inv($vars1, res, uuu$) возвращает список инвариантов, где $vars1$ — список неопределенных коэффициентов полинома, res — список уравнений, и uuu — полином. Функция находит общее решение линейной системы уравнений res, выделяет полную систему линейно-независимых решений, подставляет эти решения в полином uuu и тем самым конструирует инварианты.

```
procedure get_inv(vars1, res, uuu);
begin
  scalar s_out, sss, ind, j, j1, s1, s2, vvv, www;
  off arbvars;
  sss := solve(res,vars1);
  sss := part(sss,1);
  ind:={};
  for j:=1:length(vars1) do
  ind := append(ind, {1});
  for j:=1:length(sss) do
  begin
    for j1:=1:length(vars1) do
    if lhs(part(sss,j)) = part(vars1,j1) then
    begin
      ind := part(ind, j1) := 0;
      goto m1;
    end$;
    m1:$
  end;
  s1:={};
  for j:=1:length(ind) do
  begin
    if part(ind,j) = 1 then
    s1 := append(s1,{part(vars1,j) = 0})$;
  end;
  s_out := {};
  vvv := sub(sss,uuu);
  for j := 1:length(s1) do
  begin
    s2 := s1;
    s2 := part(s2,j) := lhs(part(s1,j)) = 1;
    www := sub(s2,vvv);
    s_out := append(s_out, {www});
  end;
  return s_out;
end;
```

В заключение данного пункта приводится пример применения описанных функций. Например, следующий фрагмент кода

```
vars := {x,y,z,u,v,w}$
gr:={{-y,x,0,-v,u,0},{-z,0,x,-w,0,u},{0,-z,y,0,-w,v}}$
uuu := func_pol(a, vars, 2)$
eqw := get_eqw(vars, gr, part(uuu,1))$
inv := get_inv(part(uuu,2), eqw, part(uuu,1));
```

выдаст список из трех инвариантов группы вращений шестимерного пространства:

$$\{x**2 + y**2 + z**2, u*x + v*y + w*z, u**2 + v**2 + w**2\}.$$

7. Приложение 1. Дифференциальные инварианты уравнений газовой динамики

Применение алгоритма описанного в пункте 4 к уравнениям трехмерной газовой динамики приводит к необозримым выражениям. Проблемы создает подгруппа вращений. С другой стороны, существует полный набор независимых полиномиальных инвариантов группы вращений в пространстве полиномов не выше третьей степени. Поэтому сначала строятся инварианты всех векторных полей кроме L^1_9, L^1_{10} и L^1_{11}. При этом для всех уравнений состояния возможен такой выбор скалярных инвариантов, что в этих пространствах инвариантов векторные поля для преобразований вращения будут иметь одинаковый вид. Однако, для этого приходится корректировать выбор инвариантов предлагаемых программой. Далее алгоритм описанный в пункте 6.3 строит 15 инвариантов группы вращений:

$$
\begin{gathered}
J^i_1 = U_i^{div}, \quad J^i_2 = (U^t_i, U^t_i), \quad J^i_3 = (U^t_i, p_i^{grad}), \quad J^i_4 = (U^t_i, \rho_i^{grad}), \\
J^i_5 = (U^t_i, U_i^{rot}), \quad J^i_6 = (p_i^{grad}, p_i^{grad}), \quad J^i_7 = (p_i^{grad}, \rho_i^{grad}), \quad J^i_8 = (\rho_i^{grad}, \rho_i^{grad}), \\
J^i_9 = (U_i^{rot}, U_i^{rot}), \quad J^i_{10} = (U_i^{rot}, p_i^{grad}), \quad J^i_{11} = (U_i^{rot}, \rho_i^{grad}), \\
J^i_{12} = (\bar{u}^x_i)^2 + (\bar{u}^y_i)^2 + (\bar{u}^z_i)^2 + (\bar{v}^x_i)^2 + (\bar{v}^y_i)^2 + (\bar{v}^z_i)^2 + (\bar{w}^x_i)^2 + (\bar{w}^y_i)^2 + (\bar{w}^z)^2_i, \\
J^i_{13} = \det(U^X_i), \quad J^i_{14} = (p_i^{grad}, U^X_i U^t_i), \quad J^i_{15} = (\rho_i^{grad}, U^X_i U^t_i).
\end{gathered}
\tag{8}
$$

Здесь через $(\cdot\,,\cdot)$ обозначается скалярное произведение трехмерных векторов, а величины стоящие в правых частях определяются в следующих девяти пунктах, при этом номер $i = 0$ соответствует общему виду функции состояния, а $i = 1, 2, \ldots, 8$ соответствуют номерам функций состояния перечисленным в таблице пункта 3.

7.0. Функция состояния общего вида

Подстановка следующих величин в формулы (8) дает первых 15 инвариантов.

$$
U^t_0 = \frac{1}{\rho^t} \begin{pmatrix} u_t + uu_x + vu_y + wu_z \\ v_t + uv_x + vv_y + wv_z \\ w_t + uw_x + vw_y + ww_z \end{pmatrix},
$$

$$
U_0^{div} = \frac{1}{\rho^t}(u_x + v_y + w_z),
$$

$$
U_0^{rot} = \frac{1}{\rho^t} \begin{pmatrix} w_y - v_z \\ u_z - w_x \\ v_x - w_y \end{pmatrix},
$$

$$
U^X_0 = \begin{pmatrix} \bar{u}_x & \bar{u}_y & \bar{u}_z \\ \bar{v}_x & \bar{v}_y & \bar{v}_z \\ \bar{w}_x & \bar{w}_y & \bar{w}_z \end{pmatrix} = \frac{1}{\rho^t} \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix},
$$

$$
p_0^{grad} = \frac{1}{\rho^t} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}, \quad \rho_0^{grad} = \frac{1}{\rho^t} \begin{pmatrix} \rho_x \\ \rho_y \\ \rho_z \end{pmatrix},
$$

где

$$
\rho^t = \rho_t + u\rho_x + v\rho_y + w\rho_z.
$$

Дополнительные инварианты:

$$J^0_{16} = \frac{1}{\rho^t}(p_t + up_x + vp_y + wp_z), \quad J^0_{17} = \rho, \quad J^0_{18} = p. \tag{9}$$

7.1. $A = pf(p^{1+2\gamma}\rho^{-1})$

Подстановка следующих величин в формулы (8) дает первых 15 инвариантов.

$$U^t_1 = p^{3\gamma+1}U^t_0, \quad U^{div}_1 = p^{2\gamma+1}U^{div}_0, \quad U^{rot}_1 = p^{2\gamma+1}U^{rot}_0,$$
$$U^X_1 = p^{2\gamma+1}U^X_0, \quad p^{grad}_1 = p^{\gamma}p^{grad}_0, \quad \rho^{grad}_1 = p^{-\gamma}\rho^{grad}_0.$$

Дополнительные инварианты:

$$J^1_{16} = p^{2\gamma}p^t_0, \quad J^1_{17} = \rho p^{-(2\gamma+1)}.$$

7.2. $A = f(\rho e^{-2p})$

Подстановка следующих величин в формулы (8) дает первых 15 инвариантов.

$$U^t_2 = e^{3p}U^t_0, \quad U^{div}_2 = e^{2p}U^{div}_0, \quad U^{rot}_2 = e^{2p}U^{rot}_0,$$
$$U^X_2 = e^{2p}U^X_0, \quad p^{grad}_2 = e^{p}p^{grad}_0, \quad \rho^{grad}_2 = e^{-p}\rho^{grad}_0.$$

Дополнительные инварианты:

$$J^2_{16} = e^{2p}p^t_0, \quad J^2_{17} = \rho e^{-2p}.$$

7.3. $A = f(\rho)$

Подстановка следующих величин в формулы (8) дает первых 15 инвариантов.

$$U^t_3 = U^t_0, \quad U^{div}_3 = U^{div}_0, \quad U^{rot}_3 = U^{rot}_0,$$
$$U^X_3 = U^X_0, \quad p^{grad}_3 = p^{grad}_0, \quad \rho^{grad}_3 = \rho^{grad}_0.$$

Дополнительные инварианты:

$$J^3_{16} = p^t_0, \quad J^3_{17} = \rho.$$

7.4. $A = f(p)$

Подстановка следующих величин в формулы (8) дает первых 15 инвариантов.

$$U^t_4 = \rho^{3/2}U^t_0, \quad U^{div}_4 = \rho U^{div}_0, \quad U^{rot}_4 = \rho U^{rot}_0,$$
$$U^X_4 = \rho U^X_0, \quad p^{grad}_4 = \rho^{1/2}p^{grad}_0, \quad \rho^{grad}_4 = \rho^{-1/2}\rho^{grad}_0.$$

Дополнительные инварианты:

$$J^4_{16} = \rho p^t_0, \quad J^4_{17} = p.$$

7.5. $A = A_0\rho^{\gamma}, \quad \gamma \neq 0$

Подстановка следующих величин в формулы (8) дает первых 15 инвариантов.

$$U_5^t = \rho^{(3-\gamma)/2} U_0^t, \quad U_5^{div} = \rho U_0^{div}, \quad U_5^{rot} = \rho U_0^{rot},$$

$$U_5^X = \rho U_0^X, \quad p_5^{grad} = \rho^{(1-\gamma)/2} p_0^{grad}, \quad \rho_5^{grad} = \rho^{(\gamma-1)/2} \rho_0^{grad}.$$

Дополнительный инвариант:

$$J_{16}^5 = \rho p_0^t.$$

7.6. $A = \gamma p, \quad \gamma \neq 0$

Подстановка следующих величин в формулы (8) дает первых 15 инвариантов.

$$U_6^t = \rho^{3/2} p^{-1/2} U_0^t, \quad U_6^{div} = \rho U_0^{div}, \quad U_6^{rot} = \rho U_0^{rot},$$

$$U_6^X = \rho U_0^X, \quad p_6^{grad} = \rho^{1/2} p^{-1/2} p_0^{grad}, \quad \rho_6^{grad} = \rho^{-1/2} p^{1/2} \rho_0^{grad}.$$

Дополнительный инвариант:

$$J_{16}^6 = \rho p^{-1} p_0^t.$$

7.7. $A = 1$

Подстановка следующих величин в формулы (8) дает первых 15 инвариантов.

$$U_7^t = \rho^{3/2} U_0^t, \quad U_7^{div} = \rho U_0^{div}, \quad U_7^{rot} = \rho U_0^{rot},$$

$$U_7^X = \rho U_0^X, \quad p_7^{grad} = \rho^{1/2} p_0^{grad}, \quad \rho_7^{grad} = \rho^{-1/2} \rho_0^{grad}.$$

Дополнительный инвариант:

$$J_{16}^7 = \rho p_0^t.$$

7.8. $A = 5/3p$

Подстановка следующих величин в формулы (8) дает все 15 инвариантов.

$$U_8^t = 3\rho^{3/2} p^{1/2} B U_0^t, \quad U_8^{div} = 3pB(\rho U_0^{div} + 1), \quad U_8^{rot} = 3\rho p B U_0^{rot},$$

$$U_8^X = 3pB(\rho U_0^X + E), \quad p_8^{grad} = 3\rho^{1/2} p^{1/2} B p_0^{grad}, \quad \rho_8^{grad} = 3\rho^{-1/2} p^{3/2} B \rho_0^{grad},$$

где E — единичная матрица, а

$$B = \frac{1}{3\rho J_{16}^0 - 5p}.$$

8. Приложение 2. Пример использования программы

В данном разделе приводится код пригодный для вычисления дифференциальных инвариантов группы симметрий уравнений газовой динамики для различных функций состояния. Приводятся входные и выходные данные для случая функции состояния общего вида.

Файл «gaz_3_0.t» с входными данными:

```
indep := {b,x,y,z};
dep   := {u,v,w,p,q};
gr:={{1,0,0,0,0,0,0,0,0},{0,1,0,0,0,0,0,0,0},{0,0,1,0,0,0,0,0,0},
{0,0,0,1,0,0,0,0,0},{0,b,0,0,1,0,0,0,0},{0,0,b,0,0,1,0,0,0},
{0,0,0,b,0,0,1,0,0},{b,x,y,z,0,0,0,0,0},
{0,-y,x,0,-v,u,0,0,0},{0,-z,0,x,-w,0,u,0,0},{0,0,-z,y,0,-w,v,0,0}}$
end;
```

Здесь символ «b» используется в качестве переменной времени (идентификатор «t» в «Reduce» занят), а символ «q» — в качестве плотности. Входные данные остальных алгебр аналогичны — в список *gr* после восьмого элемента добавляются компоненты соответствующих полей.

Далее листинг программы перемежаемый комментариями. В следующем фрагменте кода чтение файлов с функциями описанными в разделах 6.1 и 6.2, а также файла с описанием алгебры.

```
in "prol_ind.t";
in "solv_ode.t";
variant := 0;
begin
  if variant = 1 then goto m1;
  if variant = 2 then goto m2;
  if variant = 3 then goto m3;
  if variant = 4 then goto m4;
  if variant = 5 then goto m5;
  if variant = 6 then goto m6;
  if variant = 7 then goto m7;
  if variant = 8 then goto m8;
  in "gaz_3_0.t";
  goto m9;
  m1:; in "gaz_3_1.t";
  goto m9;
  m2:; in "gaz_3_2.t";
  goto m9;
  m3:; in "gaz_3_3.t";
  goto m9;
  m4:; in "gaz_3_4.t";
  goto m9;
  m5:; in "gaz_3_5.t";
  goto m9;
```

```
  m6:; in "gaz_3_6.t";
  goto m9;
  m7:; in "gaz_3_7.t";
  goto m9;
  m8:; in "gaz_3_8.t";
  m9:;
end;
```

В следующем фрагменте кода строятся операторы полного дифференцирования и продолжение алгебры до первого порядка.

```
n := length(indep);
l_ind := do_ind(1, n);
vars:=prol_vars(indep, dep, l_ind);
full_d:= full_dif(indep, dep, l_ind);
gr1 := {};
for j := 1:length(gr) do
begin
 s1 := prol_al(part(gr, j), indep, dep, l_ind, vars, full_d);
 gr1 := append(gr1,{s1});
end;
```

Здесь производится поиск всех векторных полей, которые не требуется «распрямлять» (например, поля соответствующие преобразованиям переноса) и производится проекция остальных векторных полей и переменных на соответствующее подпространство.

Этот этап можно было бы опустить, но тогда идентификаторы переменных в выходных данных были бы, как минимум, на четыре символа длиннее и время счета несколько больше.

```
ss    := test_1_all(vars, gr1)$
vars1 := part(ss,1);
gr1   := part(ss,2);
```

Здесь начинается цикл по всем векторным полям, кроме последних трех. Сначала выбирается векторное поле с компонентами линейно зависящими от координат. Дело в том, что даже, если изначально все векторные поля этому условию удовлетворяли, то в процессе проецирования на подпространства инвариантов условие может нарушиться. А реализованная здесь функция решения системы обыкновенных дифференциальных уравнений требует эту линейность.

```
r3 := {};
j := 0;
while length(gr1) > 3 do
begin
  j := j + 1;
  j0 := 0;
  for j1 := 1:length(gr1) do
```

```
if test_linear(vars1, part(gr1,j1)) = 1 then
begin
  j0 := j1;
  goto test_l;
end;
test_l:;
if j0 = 0 then
begin
  write "non linear system";
  goto toend;
end;
```

Далее происходит построение для выбранного поля системы обыкновенных дифференциальных уравнений, расщепление этой системы на независимые подсистемы, решение системы и выбор исключаемой переменной.

```
equ := split_ode(vars1, part(gr1,j0));
r := solve_ode(equ, tt);
im_list := get_impart_l(inf_jordan);
o_var := get_out_var(equ,r,tt);
```

Здесь производится корректировка выбора исключаемой переменной. Цель корректировки в обеспечении одинакового вида векторных полей для преобразований вращения в новых переменных.

```
if variant = 0 then
begin
  if j = 4 then
  begin
    o_var:={2,9,{qb1111=1},{e**tt=1/qb111}};
  end;
end;
if variant = 1 then
begin
  if j = 4 then
  begin
    o_var:={2,7,{qb1111=1},{e**tt=1/qb111}};
  end
  else
  if j = 5 then
  begin
    o_var:={2,21,{p11111=1},{e**tt=p1111,e**(gam*tt)=p1111**gam}};
  end;
end;
if variant = 2 then
begin
  if j = 4 then
  begin
```

```
    o_var:={2,7,{qb1111=1},{e**tt=1/qb111}};
  end
  else
  if j = 5 then
  begin
    o_var:={2,21,{p11111=0},{tt=p1111}};
  end;
end;
if variant = 3 then
begin
  if j = 4 then
  begin
    o_var:={2,7,{qb1111=1},{e**tt=1/qb111}};
  end
  else
  if j = 5 then
  begin
    o_var:={2,21,{p11111=0},{tt=p1111}};
  end;
end;
if variant = 4 then
begin
  if j = 4 then
  begin
    o_var:={2,7,{qb1111=1},{e**tt=1/qb111}};
  end
  else
  if j = 5 then
  begin
    o_var:={2,20,{q11111=1},{e**tt=q1111**(1/2)}};
  end;
end;
if variant = 5 then
begin
  if j = 4 then
  begin
    o_var:={2,7,{qb1111=1},{e**tt=1/qb111}};
  end
  else
  if j = 5 then
  begin
    o_var:={2,20,{q11111=1},{e**tt=q1111**(-1/2),
            e**(gam*tt)=q1111**(-gam/2)}};
  end;
end;
if variant = 6 then
begin
```

```
  if j = 4 then
  begin
    o_var:={2,7,{qb1111=1},{e**tt=1/qb111}};
  end
  else
  if j = 5 then
  begin
    o_var:={2,20,{q11111=1},{e**tt=q1111**(1/2)}};
  end
  else
  if j = 6 then
  begin
    o_var:={2,11,{p111111=1},{e**tt=p11111,e**(tt/2)=p11111**(1/2)}};
  end;
end;
if variant = 7 then
begin
  if j = 4 then
  begin
    o_var:={2,7,{qb1111=1},{e**tt=1/qb111}};
  end
  else
  if j = 5 then
  begin
    o_var:={2,20,{q11111=1},{e**tt=q1111**(1/2)}};
  end;
end;
if variant = 8 then
begin
  if j = 4 then
  begin
    o_var:={2,7,{qb1111=1},{e**tt=1/qb111}};
  end
  else
  if j = 5 then
  begin
    o_var:={2,20,{q11111=1},{e**tt=q1111**(1/2)}};
  end
  else
  if j = 6 then
  begin
    o_var:={2,11,{p111111=1},{e**tt=p11111,e**(tt/2)=p11111**(1/2)}};
  end
  else
  if j = 7 then
  begin
    o_var:={2,5,{pb1111111=1},{e**tt=(pb111111-5/3)**(1/3)}};
```

```
    end;
  end;
```

В следующем фрагменте кода производится выражение новых переменных через предыдущие и вычисление текущих инвариантов.

```
  r1:=sub({part(o_var,3,1)}, r);
  vars2:={};
  for j1 := 1:length(vars1) do
  begin
    vvv := mkid(part(vars1,j1),1);
    if vvv neq lhs(part(o_var,3,1)) then
    vars2 := append(vars2,{vvv});
  end;
  r2 := inverse_sol(equ, r1 ,o_var, tt);
  r2 := sub(part(o_var,4),r2);
  if length(im_list) > 0 then
  begin
    s1 := {};
    for j1 := 1:length(im_list) do
    if part(im_list,j1) neq 1 then
    begin
      vvv := sub(part(o_var,4), n_cos(part(im_list,j1), tt));
      s1 := append(s1,{cos(part(im_list,j1)*tt) = vvv});
      vvv := sub(part(o_var,4), n_sin(part(im_list,j1), tt));
      s1 := append(s1,{sin(part(im_list,j1)*tt) = vvv});
    end;
    r2 := sub(s1,r2);
    r1 := sub(s1,r1);
  end;
  r3 := sub(r3,r2);
```

Далее производится проецирование оставшихся векторных полей в пространство новых переменных.

```
  let cos(tt)**2 = 1 - sin(tt)**2;
  if length(gr1) > 1 then
  begin
    vars3 := {};
    for j1 := 1:length(r2) do
    vars3 := append(vars3, {lhs(part(r2,j1))});
    gr2 := {};
    for j1 := 1:length(gr1) do
    if j1 neq j0 then
    begin
      g2 := {};
      for j2:=1:length(r2) do
      begin
```

```
        vvv := field_act(vars1, part(gr1, j1), rhs(part(r2,j2)));
        g2:=append(g2,{vvv});
      end;
      if length(part(o_var,3)) > 1 then
      g2 := sub({rhs(part(o_var,3,2)) = lhs(part(o_var,3,2))}, g2);
      g2 := sub(r1,g2);
      g2 := sub(r1,g2);
      if test_depend(g2, tt) = 1 then
      begin
        begin
          g3 := test_line_dep(g2, tt);
          gr2 := append(gr2,g3)$
        end;
      end
      else
      begin
        if test_zero(g2) = 0 then
        gr2 := append(gr2,{g2})$
      end;
    end;
    vars1 := vars3;
    gr1 := gr2;
  end;
end;
toend:;
```

В заключительном фрагменте кода производится вывод списка новых переменных, представление в новых переменных векторных полей преобразований вращения, выражения новых переменных через исходные, т.е. список инвариантов всех полей, кроме трех последних.

```
begin
  if variant = 1 then goto m1;
  if variant = 2 then goto m2;
  if variant = 3 then goto m3;
  if variant = 4 then goto m4;
  if variant = 5 then goto m5;
  if variant = 6 then goto m6;
  if variant = 7 then goto m7;
  if variant = 8 then goto m8;
  out "gaz_3_0_v1.t";
  write "vars := ", vars1,"$"$
  write "gr2 := ", gr2,"$"$
  write "r3 := ", r3,"$"$
  shut "gaz_3_0_v1.t";
  goto m9;
  m1:;
```

```
out "gaz_3_1_v1.t";
write "vars := ", vars1,"$"$
write "gr2 := ", gr2,"$"$
write "r3 := ", r3,"$"$
shut "gaz_3_1_v1.t";
goto m9;
m2:;
out "gaz_3_2_v1.t";
write "vars := ", vars1,"$"$
write "gr2 := ", gr2,"$"$
write "r3 := ", r3,"$"$
shut "gaz_3_2_v1.t";
goto m9;
m3:;
out "gaz_3_3_v1.t";
write "vars := ", vars1,"$"$
write "gr2 := ", gr2,"$"$
write "r3 := ", r3,"$"$
shut "gaz_3_3_v1.t";
goto m9;
m4:;
out "gaz_3_4_v1.t";
write "vars := ", vars1,"$"$
write "gr2 := ", gr2,"$"$
write "r3 := ", r3,"$"$
shut "gaz_3_4_v1.t";
goto m9;
m5:;
out "gaz_3_5_v1.t";
write "vars := ", vars1,"$"$
write "gr2 := ", gr2,"$"$
write "r3 := ", r3,"$"$
shut "gaz_3_5_v1.t";
goto m9;
m6:;
out "gaz_3_6_v1.t";
write "vars := ", vars1,"$"$
write "gr2 := ", gr2,"$"$
write "r3 := ", r3,"$"$
shut "gaz_3_6_v1.t";
goto m9;
m7:;
out "gaz_3_7_v1.t";
write "vars := ", vars1,"$"$
write "gr2 := ", gr2,"$"$
write "r3 := ", r3,"$"$
shut "gaz_3_7_v1.t";
```

```
  goto m9;
  m8:;
  out "gaz_3_8_v1.t";
  write "vars := ", vars1,"$"$
  write "gr2 := ", gr2,"$"$
  write "r3 := ", r3,"$"$
  shut "gaz_3_8_v1.t";
  m9:;
end;
end;
```

Здесь представлены выходные данные для случая функции состояния общего вида.

```
vars := {ub1111,uz1111,vb1111,vz1111,wb1111,wz1111,pb1111,pz1111,qz1111,
uy1111,vy1111,wy1111,py1111,qy1111,ux1111,vx1111,wx1111,px1111,qx1111,
p1111,q1111}$

gr2 := {{ - vb1111, - vz1111,ub1111,uz1111,0,0,0,0,0,ux1111 - vy1111,
uy1111 + vx1111,wx1111,px1111,qx1111, - (uy1111 + vx1111),
ux1111 - vy1111, - wy1111, - py1111, - qy1111,0,0},
{ - wb1111,ux1111 - wz1111,0,vx1111,ub1111,uz1111 + wx1111,0,px1111,
qx1111, - wy1111,0,uy1111,0,0, - (uz1111 + wx1111), - vz1111,
ux1111 - wz1111, - pz1111, - qz1111,0,0},
{0,uy1111, - wb1111,vy1111 - wz1111,vb1111,vz1111 + wy1111,0,py1111,
qy1111, - uz1111, - (vz1111 + wy1111),vy1111 - wz1111, - pz1111,
 - qz1111,0, - wx1111,vx1111,0,0,0,0}}$

r3 := {ub1111=(u*ux + ub + uy*v + uz*w)/(qb + qx*u + qy*v + qz*w),
uz1111=uz/(qb + qx*u + qy*v + qz*w),
vb1111=(u*vx + v*vy + vb + vz*w)/(qb + qx*u + qy*v + qz*w),
vz1111=vz/(qb + qx*u + qy*v + qz*w),
wb1111=(u*wx + v*wy + w*wz + wb)/(qb + qx*u + qy*v + qz*w),
wz1111=wz/(qb + qx*u + qy*v + qz*w),
pb1111=(pb + px*u + py*v + pz*w)/(qb + qx*u + qy*v + qz*w),
pz1111=pz/(qb + qx*u + qy*v + qz*w),
qz1111=qz/(qb + qx*u + qy*v + qz*w),
uy1111=uy/(qb + qx*u + qy*v + qz*w),
vy1111=vy/(qb + qx*u + qy*v + qz*w),
wy1111=wy/(qb + qx*u + qy*v + qz*w),
py1111=py/(qb + qx*u + qy*v + qz*w),
qy1111=qy/(qb + qx*u + qy*v + qz*w),
ux1111=ux/(qb + qx*u + qy*v + qz*w),
vx1111=vx/(qb + qx*u + qy*v + qz*w),
wx1111=wx/(qb + qx*u + qy*v + qz*w),
px1111=px/(qb + qx*u + qy*v + qz*w),
qx1111=qx/(qb + qx*u + qy*v + qz*w),
p1111=p,q1111=q}$
```

Применение следующего кода к спискам $vars$ и $gr2$ из предыдущих выходных данных выдаст список инвариантов. Функционально независимыми среди них будут 18 инвариантов, в качестве которых можно выбрать 15 инвариантов (8) и 3 дополнительных инварианта (9) для случая функции состояния общего вида. Функции из этого фрагмента кода описаны в разделе 6.3.

```
for j := 1:3 do
begin
  uuu := func_pol(a, vars, j);
  eqw := get_eqw(vars, gr2, part(uuu,1))$
  inv := get_inv(part(uuu,2), eqw, part(uuu,1));
  write "inv_", j, " = ", inv;
end;
```

Список литературы

[1] Овсянников Л.В. Групповой анализ дифференциальных уравнений. М.: Наука, 1978, 400 с.

[2] Hern A.C. Reduce. User's and Contributed Packges Manual Version 3.8. Santa Monica, CA and Codemist Ltd. (http://reduce-algebra.sourceforge.net), 2003, 689 p.