Коммутирующие векторные поля

В.Э. Адлер. Классические интегрируемые системы, весенний семестр 2020

Если два векторных поля коммутируют, то отвечающие им динамические системы имеют совместное решение. В этой программе мы визуализируем это утверждение на численных примерах.

---

1 Немного теории

Пусть даны две конечномерные динамические системы

(1) $u_x = a(u),\ a = (a_1,\ ...,\ a_n),$

(2) $u_y = b(u),\ \ b = (b_1,\ ...,\ b_n),$

где $u = (u_1,\ ...,\ u_n)$. Можно ли построить функцию $u(x, y)$, удовлетворяющую обеим системам? Легко понять, что, вообще говоря, это невозможно. Простейший контрпример доставляет задача восстановления скалярной функции $v(x, y)$ по её частным производным:

$$v_x = f(x, y),\quad v_y = g(x, y)$$

(чтобы привести эту задачу к виду (1), (2), нужно ввести векторы $u = (x, y, v)$, $a = (1, 0, f)$ и $b = (0, 1, g)$). Ясно, что для того, чтобы такая функция v существовала (в некоторой односвязной области (x, y), в которой обе функции f и g определены и дифференцируемы), необходимо и достаточно, чтобы выполнялось равенство перекрёстных производных $(v_x)_y = \left(v_y\right)_x$, то есть

$$f_y(x, y) = g_x(x, y).$$

Если это выполнено, то ответ существует. Если нет – не существует. Всё просто.

Для систем общего вида тоже всё просто. В (1) правая часть $a(u)$ – это *векторное поле,* заданное в фазовом пространстве переменных u или в некоторой его области U. Тогда для любой (гладкой) функции $f(u)$ определена производная по x в силу системы (1), как результат применения линейного дифференциального оператора первого порядка

$$L_a = a_1\,\partial / \partial u_1 + ... + a_n\,\partial / \partial u_n,$$

то есть

$$D_x(f) = L_a(f) = a_1\,\partial f / \partial u_1 + ... + a_n\,\partial f / \partial u_n.$$

В частности, компоненты самого векторного поля совпадают с производными от координатных функций, $a_i = L_a(u_i)$. Оператор L_a можно отождествлять с самим векторным полем a. Часто, когда говорят о векторных полях, имеют в виду именно отвечающие им дифференцирования.

Критерий существования общего совместного решения систем (1), (2) заключается в перестановочности соответствующих операторов. Определим коммутатор двух векторных полей как оператор, действующий на произвольную функцию f по правилу

$$[L_a, L_b]\,(f) = L_a\,L_b(f) - L_b\,L_a(f).$$

Отметим, что произведение $L_a\,L_b$ это дифференциальный оператор второго порядка, но в коммутаторе все вторые производные сокращаются, так что он снова является векторным полем (то есть, векторные поля образуют алгебру относительно операции коммутатора. Это важный пример алгебры Ли). Явная формула для коммутатора имеет вид

(3) $$[L_a, L_b] = (L_a(b_1) - L_b(a_1))\,\partial / \partial u_1 + ... + (L_a(b_n) - L_b(a_n))\,\partial / \partial u_n.$$

Если $u(x, y)$ есть некоторое *частное* совместное решение систем (1), (2), то условие совпадения перекрёстных производных $(u_x)_y = \left(u_y\right)_x$ означает, что должно быть $[L_a, L_b]\,(u_i) = 0$ для любой

компоненты u. То есть, на рассматриваемом решении (когда мы заменяем везде u на заданную функцию $u(x, y)$) должно выполняться равенство

(4) $[L_a, L_b] = 0.$

Если же мы хотим, чтобы системы (1), (2) имели *общее* совместное решение, при каких угодно начальных условиях, то равенство (4) должно выполняться тождественно по u.

Перейдем к экспериментам.

2 Пара линейных систем с $[A, B] = 0$

Рассмотрим такую пару:

$$u_x = a = A\,u, \quad u_y = b = B\,u,$$

где A и B постоянные матрицы. В этом случае

$$\left[D_x, D_y\right](u) = D_x(B\,u) - D_y(A\,u) = [B, A]\,u.$$

Таким образом, в данном случае коммутативность векторных полей это то же самое, что коммутативность матриц. Линейные уравнения можно решать аналитически, но мы не будем этого делать, ограничимся численной проверкой. Желающие могут вывести формулу для общего совместного решения в следующем примере.

Определим функцию

nsol[A,u0,{x0,x1}] – численное решение задачи Коши $u_x = A\,u,\ u(x_0) = u_0$ при $x \in [x_0, x_1]$.

In[1]:=
```
nsol[{{a_, b_}, {c_, d_}}, {u10_, u20_}, {x0_, x1_}] :=
 NDSolve[
   {u1'[x] == a u1[x] + b u2[x],
    u2'[x] == c u1[x] + d u2[x],
    u1[x0] == u10,
    u2[x0] == u20},
   {u1, u2},
   {x, x0, x1}][[1]]
```

Зададим конкретные коммутирующие матрицы и начальное условие.

In[2]:=
```
A = {{1, 3},
    {-3, 1}};

B = {{1, 0},
    {0, 1}};

A.B - B.A  (* коммутатор матриц *)

u0 = {1, 1};

x0 = y0 = 0;
x1 = y1 = 2;
δ = ϵ = 0.2;
```

Out[4]= {{0, 0}, {0, 0}}

Решим задачу Коши для первой системы от $x = x_0$ до x_1.
Значения u в промежуточных точках, с некоторым интервалом δ, примем в качестве начальных условий для второй системы.

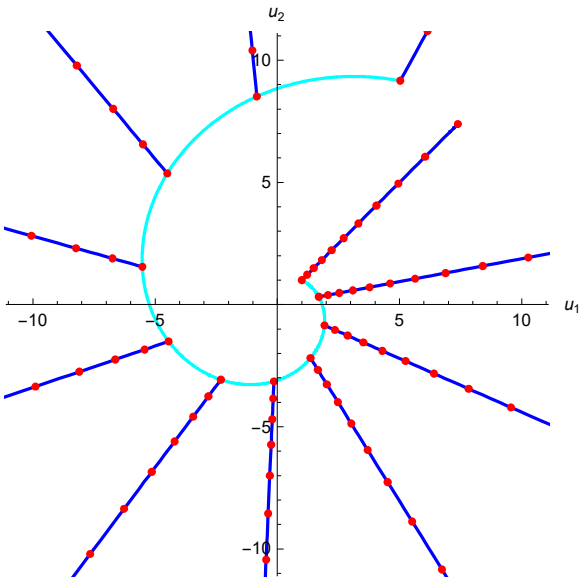
Решим соответствующие задачи Коши для второй системы, от $y = y_0$ до y_1.

Построим фазовые траектории решений и на каждой из них отметим точки с интервалом ϵ.

(Вместо фазовых траекторий можно с тем же успехом рисовать интегральные кривые. Правда, так как у нас вектор имеет размерность 2 и есть еще 2 переменные x, y, то получатся кривые в 4-мерном пространстве. Поэтому, удобнее изображать решение в виде пары графиков для двух функций $u_1(x, y)$, $u_2(x, y)$. Как это делать показано в разделе 4).

In[9]:=
```
solA = nsol[A, u0, {x0, x1}];

solBA = Table[nsol[B, {u1[x0 + i δ], u2[x0 + i δ]} /. solA, {y0, y1}], {i, 0, (x1 - x0) / δ}];

pointsBA = Table[{u1[y0 + j ϵ], u2[y0 + j ϵ]} /. solBA[[i + 1]],
    {i, 0, (x1 - x0) / δ}, {j, 0, (y1 - y0) / ϵ}];

grBA = Show[
   {ParametricPlot[{u1[x] /. solA, u2[x] /. solA}, {x, x0, x1}, PlotStyle → Cyan],
    ParametricPlot[
     Table[{u1[y] /. solBA[[i + 1]], u2[y] /. solBA[[i + 1]]}, {i, 0, (x1 - x0) / δ}],
     {y, y0, y1}, PlotStyle → Blue],
    Graphics[{Red, PointSize[0.015], Point[#] & /@ pointsBA}]
   },
   PlotRange → 10 {{-1, 1}, {-1, 1}},
   AspectRatio → Automatic,
   AxesOrigin → {0, 0},
   AxesLabel → {"u₁", "u₂"},
   ImageSize → 300
  ]
```

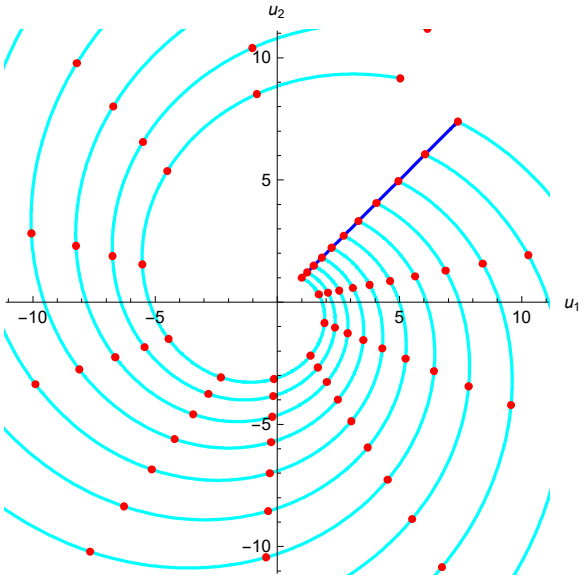
Out[12]=



Сделаем все то же самое в другом порядке.

In[13]:=
```
solB = nsol[B, u0, {x0, x1}];

solAB = Table[nsol[A, {u1[y0 + j ϵ], u2[y0 + j ϵ]} /. solB, {x0, x1}], {j, 0, (y1 - y0) / ϵ}];

pointsAB = Table[{u1[x0 + i δ], u2[x0 + i δ]} /. solAB[[j + 1]],
    {i, 0, (x1 - x0) / δ}, {j, 0, (y1 - y0) / ϵ}];

grAB = Show[
   {ParametricPlot[{u1[y] /. solB, u2[y] /. solB}, {y, y0, y1}, PlotStyle → Blue],
    ParametricPlot[
     Table[{u1[x] /. solAB[[j + 1]], u2[x] /. solAB[[j + 1]]}, {j, 0, (y1 - y0) / ϵ}],
     {x, x0, x1}, PlotStyle → Cyan],
    Graphics[{Red, PointSize[0.015], Point[#] & /@ pointsAB}]
   },
   PlotRange → 10 {{-1, 1}, {-1, 1}},
   AspectRatio → Automatic,
   AxesOrigin → {0, 0},
   AxesLabel → {"u1", "u2"},
   ImageSize → 300
  ]
```

Out[16]=

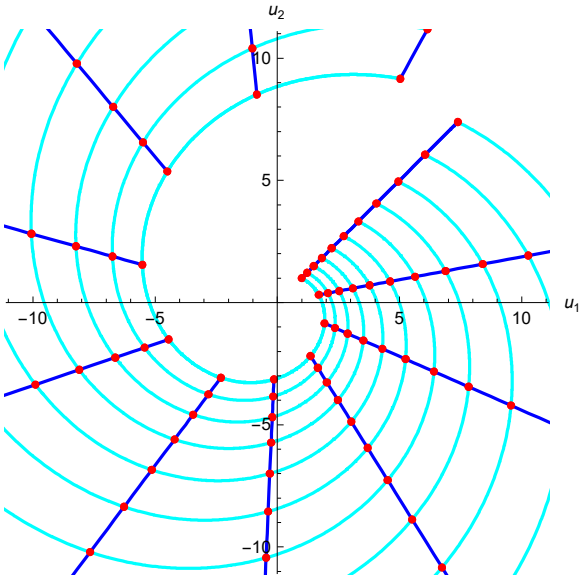


Убедимся, что отмеченные точки совпадают, в пределах точности вычислений, и совместим графики.

In[17]:=
```
Max[Abs[Flatten[pointsBA - pointsAB]]]
Show[grAB, grBA]
```

Out[17]= 0.0000349034

Out[18]=



Точки, вычисленные двумя разными способами, совпадают. Можно сначала сдвинуться на δ вдоль фазовой траектории первой системы и затем на ϵ вдоль фазовой траектории второй. Результат будет тот же, если эти сдвиги переставить. Это и означает совместность двух систем.

3 Пара линейных систем с $[A, B] \neq 0$

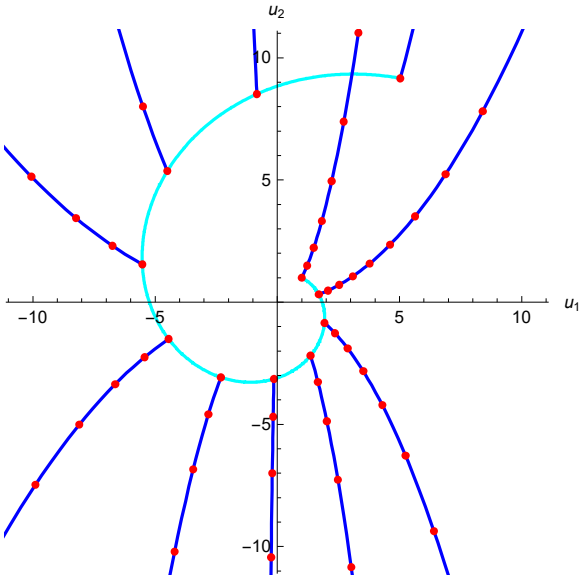
Скопируем предыдущий раздел и изменим матрицы так, чтобы они не коммутировали.

In[19]:=
```
A = {{1, 3},
    {-3, 1}};

B = {{1, 0},
    {0, 2}};

A.B - B.A (* коммутатор матриц *)

u0 = {1, 1};

x0 = y0 = 0;
x1 = y1 = 2;
δ = ϵ = 0.2;
```

Out[21]= {{0, 3}, {3, 0}}

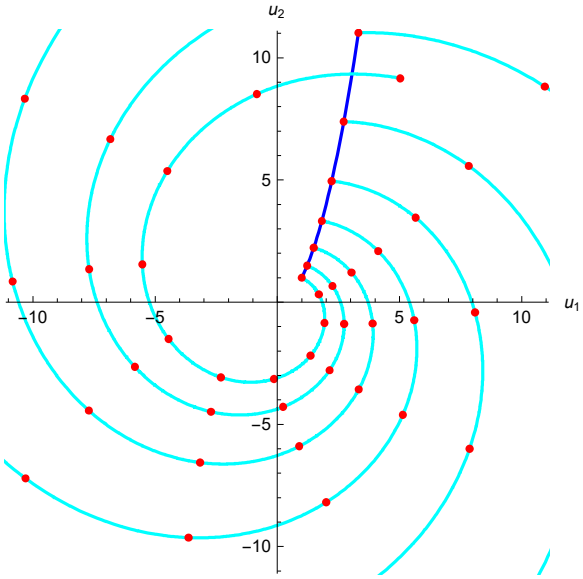
In[26]:=
```
solA = nsol[A, u0, {x0, x1}];

solBA = Table[nsol[B, {u1[x0 + i δ], u2[x0 + i δ]} /. solA, {y0, y1}], {i, 0, (x1 - x0) / δ}];

pointsBA = Table[{u1[y0 + j ϵ], u2[y0 + j ϵ]} /. solBA[[i + 1]],
    {i, 0, (x1 - x0) / δ}, {j, 0, (y1 - y0) / ϵ}];

grBA = Show[
   {ParametricPlot[{u1[x] /. solA, u2[x] /. solA}, {x, x0, x1}, PlotStyle → Cyan],
    ParametricPlot[
     Table[{u1[y] /. solBA[[i + 1]], u2[y] /. solBA[[i + 1]]}, {i, 0, (x1 - x0) / δ}],
     {y, y0, y1}, PlotStyle → Blue],
    Graphics[{Red, PointSize[0.015], Point[#] & /@ pointsBA}]
   },
   PlotRange → 10 {{-1, 1}, {-1, 1}},
   AspectRatio → Automatic,
   AxesOrigin → {0, 0},
   AxesLabel → {"u1", "u2"},
   ImageSize → 300
  ]
```

Out[29]=



In[30]:=
```
solB = nsol[B, u0, {x0, x1}];

solAB = Table[nsol[A, {u1[y0 + j ϵ], u2[y0 + j ϵ]} /. solB, {x0, x1}], {j, 0, (y1 - y0) / ϵ}];

pointsAB = Table[{u1[x0 + i δ], u2[x0 + i δ]} /. solAB[[j + 1]],
    {i, 0, (x1 - x0) / δ}, {j, 0, (y1 - y0) / ϵ}];

grAB = Show[
   {ParametricPlot[{u1[y] /. solB, u2[y] /. solB}, {y, y0, y1}, PlotStyle → Blue],
    ParametricPlot[
     Table[{u1[x] /. solAB[[j + 1]], u2[x] /. solAB[[j + 1]]}, {j, 0, (y1 - y0) / ϵ}],
     {x, x0, x1}, PlotStyle → Cyan],
    Graphics[{Red, PointSize[0.015], Point[#] & /@ pointsAB}]
   },
   PlotRange → 10 {{-1, 1}, {-1, 1}},
   AspectRatio → Automatic,
   AxesOrigin → {0, 0},
   AxesLabel → {"u1", "u2"},
   ImageSize → 300
  ]
```

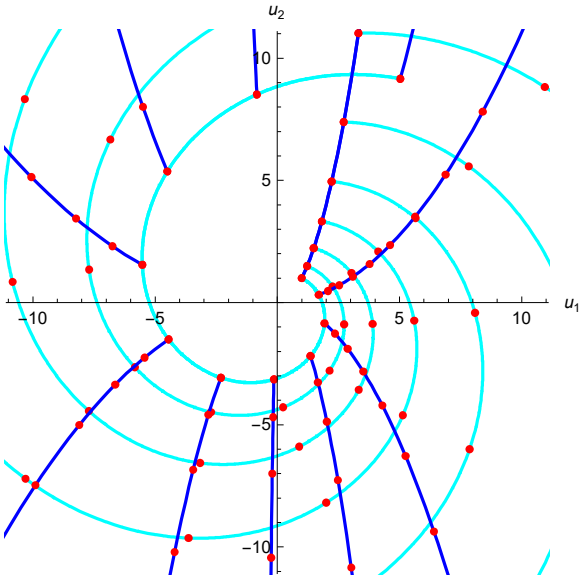
Out[33]=



In[34]:=
```
Max[Abs[Flatten[pointsBA - pointsAB]]]
Show[grAB, grBA]
```

Out[34]= 232.931

Out[35]=



Сетка расползлась, красные точки не попадают в узлы. Теперь не всё равно, в каком порядке ходить по траекториям. Системы не совместны.

4 Нелинейный пример

Здесь мы сделаем то же самое для нелинейного уравнения, и заодно проиллюстрируем ещё два момента:

– как работать в том случае, если вместо динамической системы (1) задано ОДУ порядка n;
– как выводить график решения вместо фазовых траекторий.

Задача. Пусть обозначено $u_n = \partial_x^n(u)$. Проверить, символически и численно, что ОДУ

(5) $$u_4 + 10\,u\,u_2 + 5\,u_1^2 + 10\,u^3 + c_1\left(u_2 + 3\,u^2\right) + c_2\,u + c_3 = 0,$$

где c_1, c_2, c_3 произвольные постоянные, совместно с уравнением КдФ

(6) $$u_t = u_3 + 6\,u\,u_1.$$

Если это действительно так, построить график какого-нибудь решения $u(x, t)$ удовлетворяющего обоим уравнениям.

Решение. Перепишем оба уравнения в виде динамических систем. Так как 4-я производная выражается из ОДУ (5) через младшие, то динамическими переменными являются u_0, u_1, u_2, u_3 (для единообразия будем писать u_0 вместо u). Сразу можно написать векторное поле D_x (точнее – то, как оно действует на произвольную функцию от динамических переменных):

In[36]:=
```
dx[f_] := u[1] D[f, u[0]] +
  u[2] D[f, u[1]] +
  u[3] D[f, u[2]] -
  (10 u[0] u[2] + 5 u[1]^2 + 10 u[0]^3 + c1 (u[2] + 3 u[0]^2) + c2 u[0] + c3) D[f, u[3]]
```

С ОДУ справились. С КдФ немного посложнее. Это уравнение задаёт правило дифференцирования по t только для переменной u_0, а нам нужно ещё для u_1, u_2, u_3. Значит, нужно *продолжить* дифференцирование D_t на эти производные, пользуясь тем, что D_x уже определено и полагая, по

определению, что $D_t(u_k) = D_x^k(u_t)$. Имеем:

In[37]:=
```
ut = u[3] + 6 u[0] u[1];

dt[f_] := ut D[f, u[0]] +
  dx[ut] D[f, u[1]] +
  dx[dx[ut]] D[f, u[2]] +
  dx[dx[dx[ut]]] D[f, u[3]]
```

Можно посмотреть, что за векторное поле получилось. Вот его компоненты:

In[39]:=
```
dt[u[0]]
Expand[dt[u[1]]]
Expand[dt[u[2]]]
Expand[dt[u[3]]]
```

Out[39]= $6\,u[0]\,u[1] + u[3]$

Out[40]= $-c3 - c2\,u[0] - 3\,c1\,u[0]^2 - 10\,u[0]^3 + u[1]^2 - c1\,u[2] - 4\,u[0]\,u[2]$

Out[41]= $-c2\,u[1] - 6\,c1\,u[0]\,u[1] - 30\,u[0]^2\,u[1] - 2\,u[1]\,u[2] - c1\,u[3] - 4\,u[0]\,u[3]$

Out[42]= $c1\,c3 + c1\,c2\,u[0] + 4\,c3\,u[0] + 3\,c1^2\,u[0]^2 + 4\,c2\,u[0]^2 + 22\,c1\,u[0]^3 + 40\,u[0]^4 - c1\,u[1]^2 -$
$40\,u[0]\,u[1]^2 + c1^2\,u[2] - c2\,u[2] + 8\,c1\,u[0]\,u[2] + 10\,u[0]^2\,u[2] - 2\,u[2]^2 - 6\,u[1]\,u[3]$

Итак, построили два векторных поля. Считаем коммутатор. Для этого нужно вычислить его значения на координатных функциях (= динамических переменных), то есть, $[D_x, D_t]\,(u_k),\ k = 0, 1, 2, 3$.

In[43]:=
```
Table[Expand[dx[dt[u[k]]] - dt[dx[u[k]]]], {k, 0, 3}]
```

Out[43]= $\{0, 0, 0, 0\}$

Всё сошлось. На самом деле, ясно, что равенство нулю коммутатора на первых трёх компонентах выполняется автоматически – в силу построения дифференцирования D_t по формуле $D_t(u_k) = D_x^k(u_t)$. Для примера, испортим что-то в уравнениях. Тогда коммутативность нарушится, но только на последней компоненте. Так что, можно было бы проверять только её, но лишний контроль не мешает.

In[44]:=
```
(* портим ut *)
ut = u[3] + 66 u[0] u[1];
Table[Expand[dx[dt[u[k]]] - dt[dx[u[k]]]], {k, 0, 3}]

(* откатываем назад *)
ut = u[3] + 6 u[0] u[1];
Table[Expand[dx[dt[u[k]]] - dt[dx[u[k]]]], {k, 0, 3}]
```

Out[45]= $\{0, 0, 0, -300\,c3\,u[1] - 300\,c2\,u[0]\,u[1] - 900\,c1\,u[0]^2\,u[1] -$
$3000\,u[0]^3\,u[1] - 900\,u[1]^3 - 120\,c1\,u[1]\,u[2] - 1200\,u[0]\,u[1]\,u[2] + 600\,u[2]\,u[3]\}$

Out[47]= $\{0, 0, 0, 0\}$

Переходим к численному счёту. Определяем решение задачи Коши по x.

In[48]:=
```
nsolx[{u00_, u10_, u20_, u30_}, {c1_, c2_, c3_}, {x0_, x1_}] :=
 NDSolve[
   {u[0]'[x] == u[1][x],
    u[1]'[x] == u[2][x],
    u[2]'[x] == u[3][x],
    u[3]'[x] == -(10 u[0][x] u[2][x] +
        5 u[1][x]^2 + 10 u[0][x]^3 + c1 (u[2][x] + 3 u[0][x]^2) + c2 u[0][x] + c3),
    u[0][x0] == u00,
    u[1][x0] == u10,
    u[2][x0] == u20,
    u[3][x0] == u30},
   {u[0], u[1], u[2], u[3]},
   {x, x0, x1}][[1]]
```

Определяем решение задачи Коши по t. Нам нужно добавить аргумент t к тем обозначениям, что у нас были. Сделаем это в следующей ячейке, а потом скопируем (хотя такая практика не есть хорошо. Для более серьезных задач нужен какой-то другой способ).

In[49]:=
```
dt[u[0]] /. u[k_] :> u[k][t]
Expand[dt[u[1]]] /. u[k_] :> u[k][t]
Expand[dt[u[2]]] /. u[k_] :> u[k][t]
Expand[dt[u[3]]] /. u[k_] :> u[k][t]
```

Out[49]=
```
6 u[0][t] u[1][t] + u[3][t]
```

Out[50]=
```
-c3 - c2 u[0][t] - 3 c1 u[0][t]^2 - 10 u[0][t]^3 + u[1][t]^2 - c1 u[2][t] - 4 u[0][t] u[2][t]
```

Out[51]=
```
-c2 u[1][t] - 6 c1 u[0][t] u[1][t] - 30 u[0][t]^2 u[1][t] -
 2 u[1][t] u[2][t] - c1 u[3][t] - 4 u[0][t] u[3][t]
```

Out[52]=
```
c1 c3 + c1 c2 u[0][t] + 4 c3 u[0][t] + 3 c1^2 u[0][t]^2 + 4 c2 u[0][t]^2 + 22 c1 u[0][t]^3 +
 40 u[0][t]^4 - c1 u[1][t]^2 - 40 u[0][t] u[1][t]^2 + c1^2 u[2][t] - c2 u[2][t] +
 8 c1 u[0][t] u[2][t] + 10 u[0][t]^2 u[2][t] - 2 u[2][t]^2 - 6 u[1][t] u[3][t]
```

In[53]:=
```
nsolt[{u00_, u10_, u20_, u30_}, {c1_, c2_, c3_}, {t0_, t1_}] :=
 NDSolve[
   {u[0]'[t] == 6 u[0][t] u[1][t] + u[3][t],
    u[1]'[t] ==
     -c3 - c2 u[0][t] - 3 c1 u[0][t]^2 - 10 u[0][t]^3 + u[1][t]^2 - c1 u[2][t] - 4 u[0][t] u[2][t],
    u[2]'[t] == -c2 u[1][t] - 6 c1 u[0][t] u[1][t] - 30 u[0][t]^2 u[1][t] -
      2 u[1][t] u[2][t] - c1 u[3][t] - 4 u[0][t] u[3][t],
    u[3]'[t] == c1 c3 + c1 c2 u[0][t] + 4 c3 u[0][t] + 3 c1^2 u[0][t]^2 + 4 c2 u[0][t]^2 +
      22 c1 u[0][t]^3 + 40 u[0][t]^4 - c1 u[1][t]^2 - 40 u[0][t] u[1][t]^2 + c1^2 u[2][t] -
      c2 u[2][t] + 8 c1 u[0][t] u[2][t] + 10 u[0][t]^2 u[2][t] - 2 u[2][t]^2 - 6 u[1][t] u[3][t],
    u[0][t0] == u00,
    u[1][t0] == u10,
    u[2][t0] == u20,
    u[3][t0] == u30},
   {u[0], u[1], u[2], u[3]},
   {t, t0, t1}][[1]]
```

Дальше – всё почти как в предыдущих примерах. Решаем задачу Коши для (5) от $x = x_0$ до x_1. Значения вектора (u_0, u_1, u_2, u_3) в промежуточных точках, с некоторым шагом δ, принимаем в качестве начальных условий для (6). Решаем соответствующие задачи Коши от $t = t_0$ до t_1. Потом делаем то же

самое в другом порядке, с шагом ϵ по t. Сравниваем получившиеся сетки. Если значения совпадают – значит функция $u(x, t)$ определена корректно, строим её 3D график по полученной сетке.

Следует отметить, что в этой задаче многие решения имеют полюсные особенности. Ниже, значения параметров и начальных условий подобраны так, чтобы их не было.

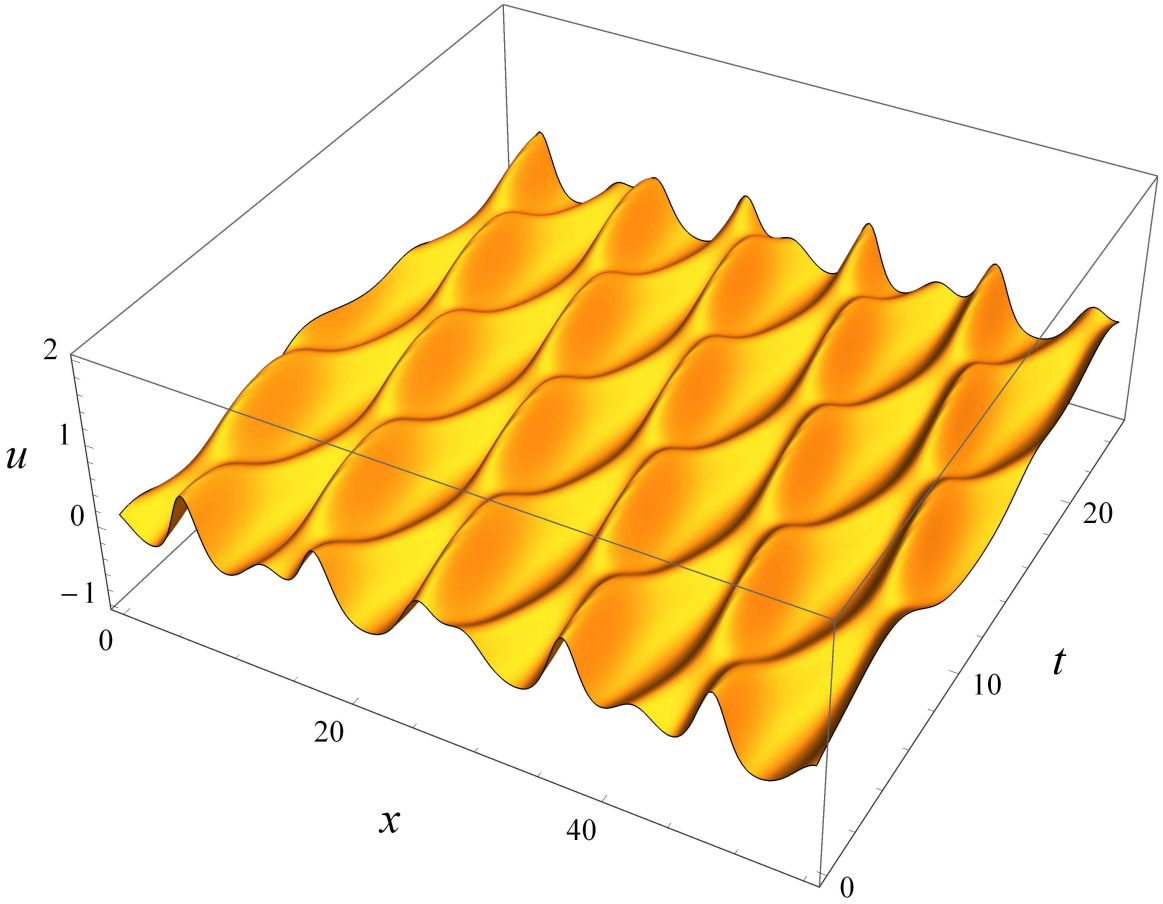
In[54]:=
```
x0 = 0;
x1 = 55;
Mx = 200; (* число точек в сетке по x *)

t0 = 0;
t1 = 25;
Mt = 200; (* число точек в сетке по t *)

cc = {0.15, -0.7, 0}; (* параметры *)

U0 = {0.1, -0.1, -0.1, 0.125};  (* начальные данные при x0, t0 *)

(* решаем систему по t; решаем систему по x для последовательности
  значений t[0]= t0, ..., t[Mt]=t1 и сохраняем результат в виде 2D массива *)

nst = nsolt[U0, cc, {t0, t1}];

nstx2 = Table[
    ns = nsolx[
      {u[0][t], u[1][t], u[2][t], u[3][t]} /. t → t0 + j (t1 - t0) / Mt /. nst, cc, {x0, x1}];
    Table[u[0][x0 + i (x1 - x0) / Mx] /. ns, {i, 0, Mx}], {j, 0, Mt}];

(* делаем то же самое, меняя ролями x и t *)
nsx = nsolx[U0, cc, {x0, x1}];

nsxt2 = Table[
    ns = nsolt[
      {u[0][x], u[1][x], u[2][x], u[3][x]} /. x → x0 + i (x1 - x0) / Mx /. nsx, cc, {t0, t1}];
    Table[u[0][t0 + j (t1 - t0) / Mt] /. ns, {j, 0, Mt}], {i, 0, Mx}];

(* сравниваем две сетки *)
Max[Abs[nstx2 - Transpose[nsxt2]]]
```

Out[62]= `0.00092204`

Значения, вычисленные двумя способами, совпали, что и подтверждает коммутативность. Теперь можно использовать полученный массив точек, чтобы построить график решения.

In[63]:=
```
ListPlot3D[nstx2,
 DataRange → {{x0, x1}, {t0, t1}},
 PlotRange → {-1, 2},
 Mesh → None,
 AxesLabel → Evaluate[Style[#, Italic, 24] & /@ {"x", "t", "u"}],
 ImageSize → 600,
 BaseStyle → {FontFamily → "Times", FontSize → 16}
]
```

Out[63]=



Пояснение. То, что у нас получилось это пример так называемого двухзонного, или двухфазного решения КдФ. Уравнение (5) это один раз проинтегрированное стационарное уравнение для *высшей симметрии* КдФ пятого порядка плюс младшие симметрии с константами c_i. Мы проверили, на примере, что такие стационарные уравнения приводят к паре коммутирующих векторных полей, что и позволяет использовать их для построения решений.