

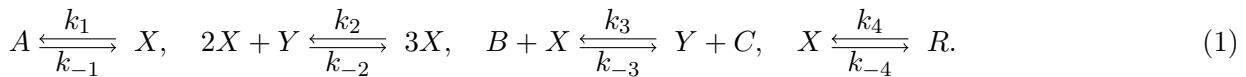
Брюсселятор

Содержание

1 Постановка задачи	1
2 Нераспределенный случай	2
2.1 Исследование на устойчивость нераспределенного случая	2
2.2 Предварительный анализ нераспределенного случая в Wolfram Mathematica	3
2.3 Анализ нераспределенного случая на С	4
3 Распределенный случай	6
3.1 Явная реализация	6
3.2 Подготовительные расчеты	8
3.3 Метод Кранка-Николсона	10
3.4 Границные условия	11
3.5 Исследование устойчивости разностной схемы	12
3.5.1 Признак максимума	12
3.5.2 Энергетический критерий устойчивости	13
3.6 Исследование на устойчивость распределенного случая	13
3.7 Неустойчивость Тьюринга	14
3.8 Реализация на С	16
3.9 Распределенный случай 2D	20
3.10 Анализ распределенного случая на Wolfram	24

1 Постановка задачи

Пусть в тонкой замкнутой трубке протекает следующая реакция:



Здесь A, B - исходные вещества, C, R - продукты реакции, а X, Y - промежуточные вещества.

Пусть конечные продукты C и R немедленно удаляются из реакционного пространства, это значит, что $k_{-3} = k_{-4} = 0$. Если субстрат A находится в избытке, то $k_{-1} = 0$. Предположим, что k_{-2} также ноль.

Соответствующие дифференциальные уравнения для промежуточных веществ X и Y запишутся:

$$\frac{\partial X}{\partial t} = k_1 A + k_2 X^2 Y - k_3 B X - k_4 X, \quad (2)$$

$$\frac{\partial Y}{\partial t} = -k_2 X^2 Y + k_3 B X. \quad (3)$$

Проиведем замену переменных:

$$X' = \frac{X}{X_0}, \quad Y' = \frac{Y}{Y_0}, \quad t' = \frac{t}{t_0}, \quad k'_i = \frac{k_i}{k_i^0}, \quad (4)$$

где

$$k_1^0 = \frac{X_0}{A_0 t_0}, \quad k_2^0 = \frac{1}{X_0 Y_0 t_0}, \quad k_3^0 = \frac{1}{B_0 t_0}, \quad k_4^0 = \frac{1}{t_0}. \quad (5)$$

X' и Y' - безразмерные концентрации веществ X, Y ; X_0, Y_0, A_0, B_0 - характерные значения концентраций веществ X, Y, A, B ; t_0 - характерное время реакций.

В безразмерных переменных система приобретет вид (штрихи опущены)

$$\frac{\partial X}{\partial t} = A + X^2 Y - (B + 1) X, \quad (6)$$

$$\frac{\partial Y}{\partial t} = -X^2 Y + B X. \quad (7)$$

Добавим в систему диффузию, получим:

$$\frac{\partial X}{\partial t} = A + X^2Y - (B + 1)X + D_x \frac{\partial^2 X}{\partial r^2}, \quad (8)$$

$$\frac{\partial Y}{\partial t} = -X^2Y + BX + D_y \frac{\partial^2 Y}{\partial r^2}. \quad (9)$$

2 Нераспределенный случай

2.1 Исследование на устойчивость нераспределенного случая

$$\begin{cases} \dot{u} = A + u^2v - (B + 1)u + D_u \Delta u, \\ \dot{v} = -u^2v + Bu + D_v \Delta v. \end{cases} \quad (10)$$

Линеаризуем эту систему:

$$\begin{cases} A + u^2v - (B + 1)u = 0, \\ u(B - uv) = 0. \end{cases} \quad (11)$$

Предположим, что коэффициенты А и В больше нуля, тогда решение этой системы:

$$\begin{cases} \bar{u} = A, \\ \bar{v} = \frac{B}{A}. \end{cases} \quad (12)$$

Линеаризуем систему вблизи этой особой точки:

$$\xi = u - \bar{u}, \quad \eta = v - \bar{v}. \quad (13)$$

$$\begin{cases} \dot{\xi} = (B - 1)\xi + A^2\eta, \\ \dot{\eta} = -B\xi - A^2\eta. \end{cases} \quad (14)$$

Найдем собственные значения матрицы коэффициентов:

$$\begin{vmatrix} B - 1 - \lambda & A^2 \\ -B & -A^2 - \lambda \end{vmatrix} = 0. \quad (15)$$

$$\lambda^2 + (A^2 + 1 - B)\lambda + A^2 = 0. \quad (16)$$

Решения этого уравнения:

$$\lambda_{1,2} = -\frac{1}{2}(A^2 + 1 - B) \pm \frac{1}{2}\sqrt{(A^2 + 1 - B)^2 - 4A^2}. \quad (17)$$

Введем обозначения: $\delta = (A^2 + 1 - B)$ и $\sigma = A^2$, тогда уравнение перепишется

$$\lambda_{1,2} = -\frac{1}{2}\delta \pm \frac{1}{2}\sqrt{\delta^2 - 4\sigma}. \quad (18)$$

Проанализируем возможные случаи:

- $\delta = 0$, тогда $\lambda_{1,2}$ чисто мнимые и решение является центром
- $\delta^2 < 4\sigma$, тогда корни мнимые и устойчивость зависит от знака δ : при $\delta > 0$ устойчивый фокус, при $\delta < 0$ неустойчивый фокус
- $\delta^2 > 4\sigma$, значит, корни действительные и простой анализ показывает, что знак $\lambda_{1,2}$ также зависит от знака δ : при $\delta > 0$ устойчивый узел, при $\delta < 0$ неустойчивый узел
- $\sigma < 0$, тогда корни действительные, но разных знаков, реализуется случай седла

$\delta = 0$ точка, в которой происходит смена характера поведения решения с устойчивости на неустойчивость, происходит бифуркация. Если $Re(\lambda_{1,2}) = 0$, а $Im(\lambda_{1,2}) \neq 0$, то происходит бифуркация Андронова-Хопфа или бифуркация рождения (исчезновения) предельного цикла. Из устойчивого фокуса при изменении параметров A, B может родиться предельный цикл.

2.2 Предварительный анализ нераспределенного случая в Wolfram Mathematica

Прежде чем программировать задачу на С или Python, проанализируем ее, используя Wolfram Mathematica.

```
Export[FileNameJoin@{NotebookDirectory[], "animation_a=1 b=1.gif"}, 
Animate[
A = 1; B = 1;
sol = NDSolve[{x'[t] == A + x[t]^2 y[t] - (B + 1) x[t], 
y'[t] == x[t]^2 y[t] + B x[t], x[0] == 1, y[0] == 2}, {x[t], 
y[t]}, {t, 0, T}];
ParametricPlot[{x[t], y[t]} /. sol, {t, 0, T}], {T, 1, 100}
], "GIF"]
```

В коде можно менять параметры A и B, а также смотреть на картинку с течением времени.

Прикрепим также зависимость $x(t)$ и $y(t)$:

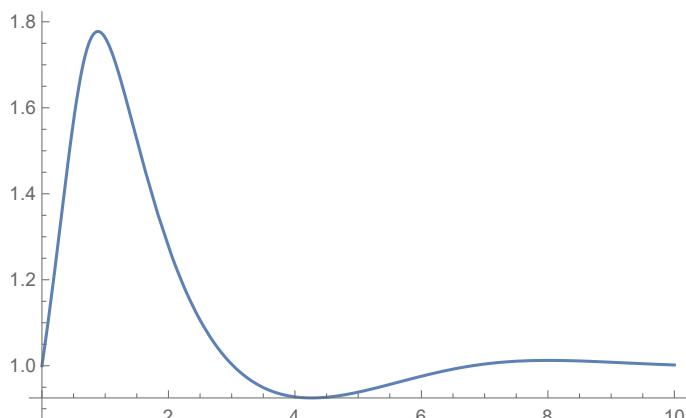


Рис. 1: Зависимость $x(t)$

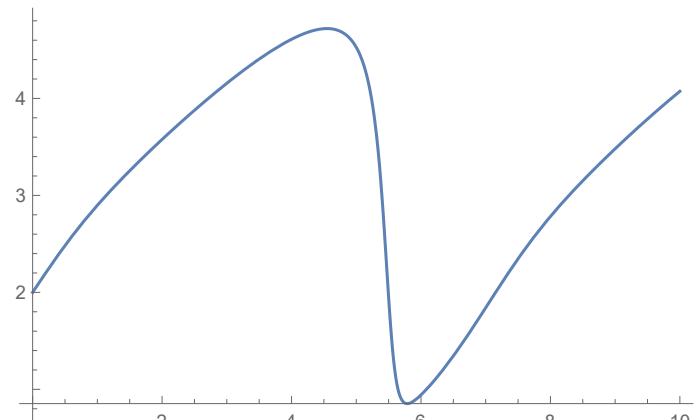


Рис. 2: Зависимость $y(t)$

Приведем характерные параметрические графики для случая $A=1, B=1$ и $A=1, B=3$. Как видно, в первом случае система устойчива, а во втором неустойчива. Это подтверждает пункт 1 отчета.

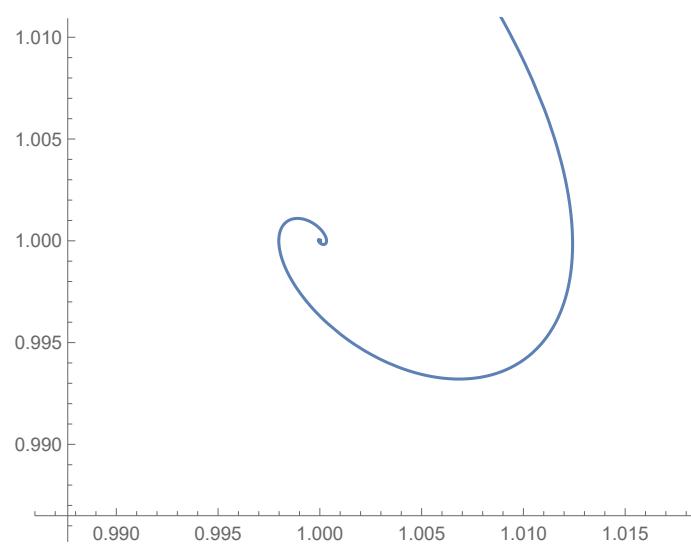


Рис. 3: A=1, B=1

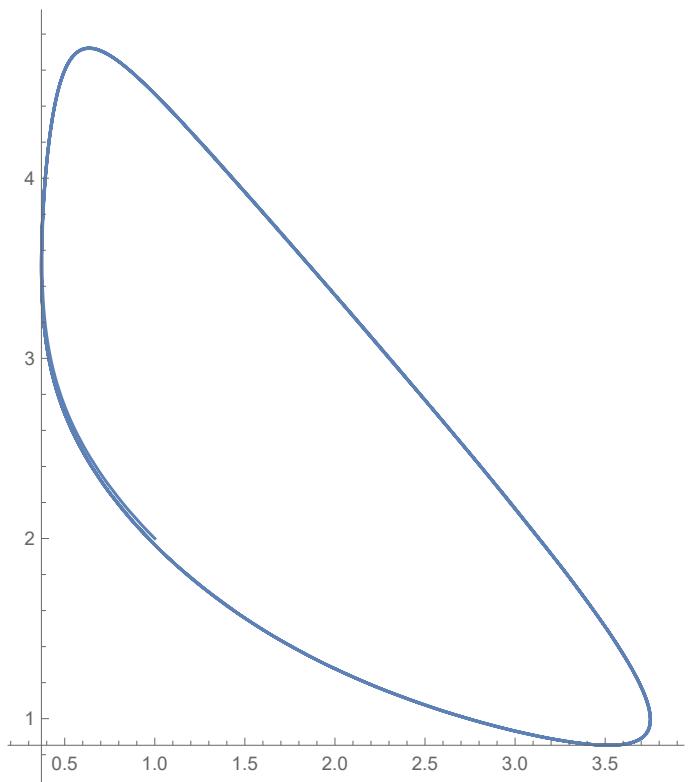


Рис. 4: A=1, B=3

2.3 Анализ нераспределенного случая на С

Будем использовать метод Рунге-Кутты.

Приведем исполняемый код:

```

1 #include <stdio.h>
2 #include<math.h>
3 #include<stdlib.h>
4
5 const float A = 1;
6 const float B = 3;
7
8 float F1( float u, float v)
9 {
10     return (A + u*u*v*(B+1)*u);
11 }
12
13 float F2( float u, float v)
14 {
15     return ( u*u*v + B*u);
16 }
17
18 void runge_kutt( float h, float* u, float* v, int N)
19 {
20 FILE* fp ;
21 fp = fopen("brus" , "w");
22
23     for( int i = 0; i < N; i++)
24     {
25         float k11 = 0, k12 = 0, k13 = 0, k14 = 0;
26         float k21 = 0, k22 = 0, k23 = 0, k24 = 0;
27         k11 = h*F1(u[ i] , v[ i]);
28         k21 = h*F2(u[ i] , v[ i]);
29         k12 = h*F1(u[ i] + k11/2, v[ i] + k21/2);
30         k22 = h*F2(u[ i] + k11/2, v[ i] + k21/2);
31

```

```

32 k13 = h*F1(u[i] + k12/2, v[i] + k22/2);
33 k23 = h*F2(u[i] + k12/2, v[i] + k22/2);
34 k14 = h*F1(u[i] + k13/2, v[i] + k23/2);
35 k24 = h*F2(u[i] + k13/2, v[i] + k23/2);
36 u[i+1] = u[i] + (k11 + 2*k12 + 2*k13 + k14)/6;
37 v[i+1] = v[i] + (k21 + 2*k22 + 2*k23 + k24)/6;
38 fprintf(fp, "%f, %f, %f \n", h, u[i], v[i]);
39 }
fclose(fp);
40 }
41
42 int main()
43 {
44 int n = 100000;
45 float u[n];
46 float v[n];
47 float t[n];
48
49 u[0] = 1;
50 v[0] = 2;
51 t[0] = 0;
52 t[100] = 20;
53 float tau = 0.001;
54 int N = (t[100] - t[0])/tau;
55 runge_kutt(tau, &u[0], &v[0], N);
56 return 0;
57 }

```

Построим три графика: $u(t)$, $v(t)$ и параметрический график u, v для двух случаев:

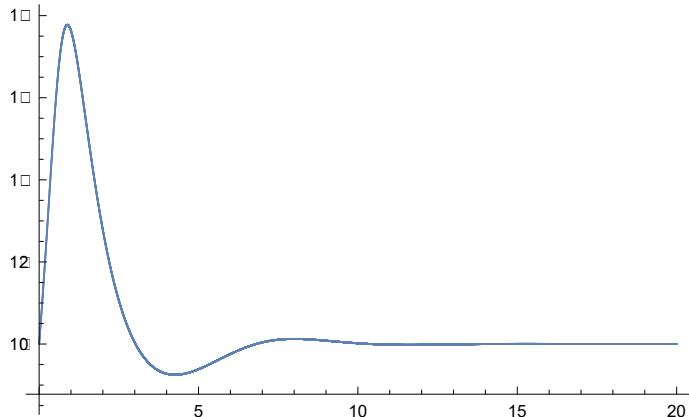


Рис. 5: $u(t)$ $A=1$, $B=11$ цвцв

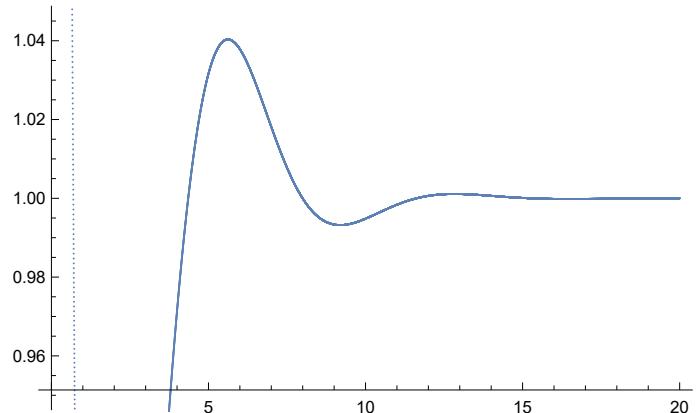


Рис. 6: $v(t)$ $A=1$, $B=1$

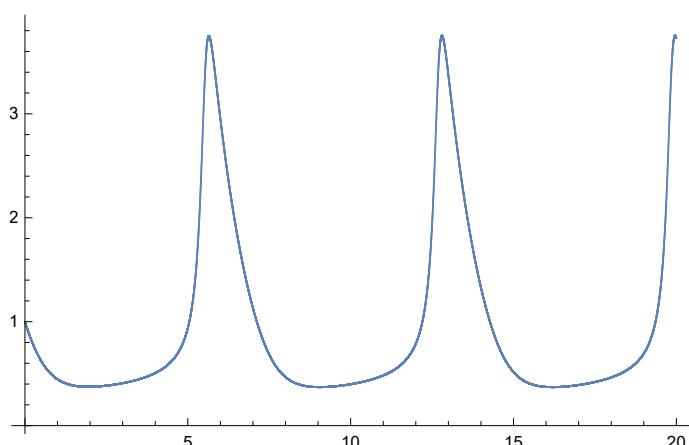


Рис. 7: $u(t)$ $A=1$, $B=3$

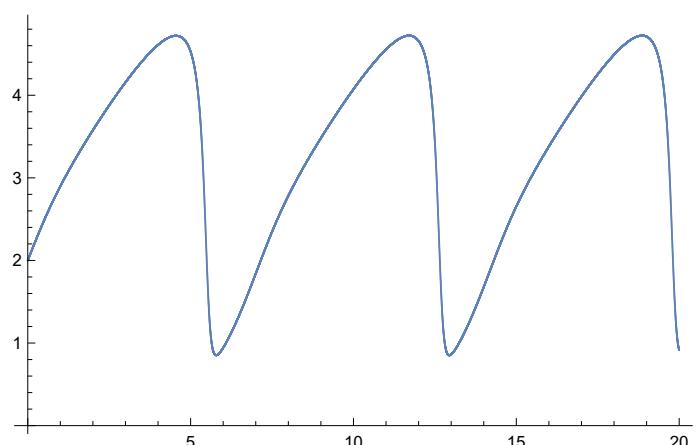


Рис. 8: $v(t)$ $A=1$, $B=3$

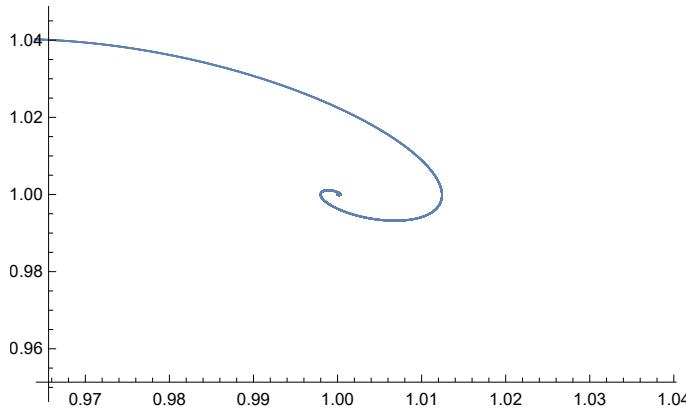


Рис. 9: $A=1, B=1$

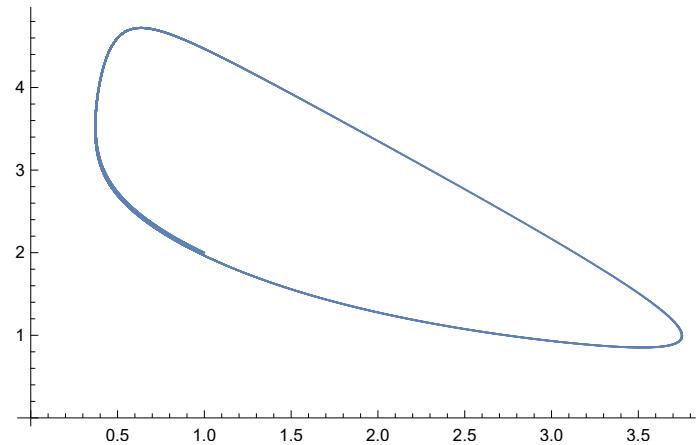


Рис. 10: $A=1, B=3$

Результаты совпадают с полученными в пункте 2.

3 Распределенный случай

3.1 Явная реализация

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 const float A = 1;
5 const float B = 1;
6 const float Du = 1;
7 const float Dv = 100;
8
9 int main(){
10
11 float a = 0;
12 float b = 1;
13 float T = 1;
14 float t0 = 0;
15 float h = 0.01;
16 float tau = 0.01;
17 int Nx = (b - a) / h;
18 int Nt = (T - t0) / tau;
19
20 float ru = Du*tau/2/h/h;
21 float rv = Dv*tau/2/h/h;
22
23 float fu[Nt+1][Nx+1];
24 float fv[Nt+1][Nx+1];
25
26 for( int j = 0; j < Nt; j++){
27     for( int i = 0; i < Nx; i++){
28         fu[j][i] = 0.;
29         fv[j][i] = 0.;
30     }
31 }
32
33 double noiseu = 0.005*(rand()%100);
34 double noisev = 0.005*(rand()%100);
35 for( int i = 0; i <= Nx; i++){
36     fv[0][i] = B/A + noisev;
37     fu[0][i] = A + noiseu;
38 }
39
40 for( int j = 1; j <= Nt; j++){

```

```

42 for( int i = 1; i <Nx; i++){
43   fu[j][i] = fu[j-1][i] + tau*(A+fu[j-1][i]*fu[j-1][i]*fv[j-1][i] - (B+1)*fu[j-1][i] + Du/h/h*(fu[j-1][i+1]+fu[j-1][i-1] 2* fu[j-1][i]));
44   fv[j][i] = fv[j-1][i] + tau*(- fu[j-1][i]*fu[j-1][i]*fv[j-1][i] + B*fu[j-1][i] + Dv/h/h*(fv[j-1][i] +1)+fv[j-1][i-1] 2* fv[j-1][i]));
45 }
46   fu[j][0] = fu[j][1];
47   fv[j][0] = fv[j][1];
48   fv[j][Nx] = fv[j][Nx-1];
49   fu[j][Nx] = fu[j][Nx-1];
50 }
51
52 FILE* fp ;
53 fp = fopen("brus.dat", "w");
54 for( int j = 0; j <= Nt; j++){
55   for( int i = 0; i <= Nx; i++){
56     fprintf(fp, "%f %f %f %f \n", j*tau, i*h, fu[j][i], fv[j][i]);
57   }
58 }
59 fclose(fp);
60 return 0;
61 }

```

Приведем результат выполнения программы:

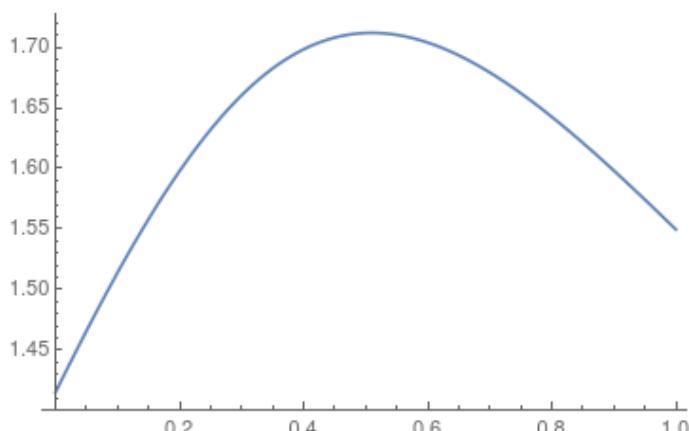


Рис. 11: Явная схема $u(t)$, $A=1$, $B=1$, $Du = 1$, $Dv = 100$

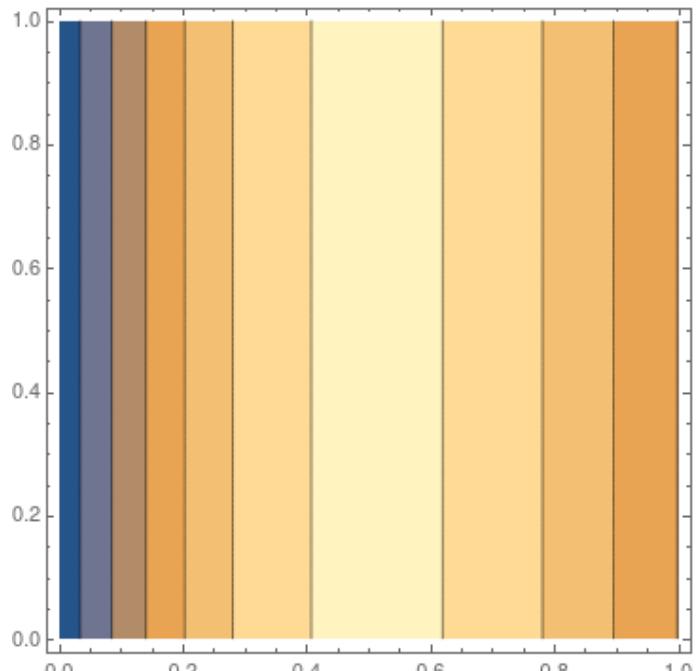


Рис. 12: Явная схема $u(t,x)$ $A=1$, $B=1$, $Du = 1$, $Dv = 100$

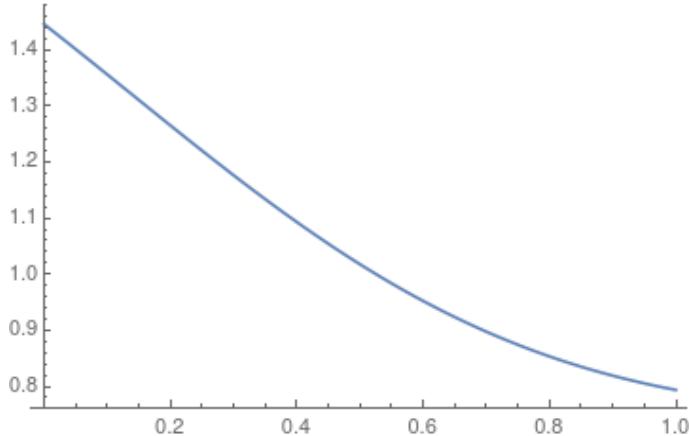


Рис. 13: Явная схема $v(t)$, $A=1$, $B=1$, $D_u = 1$, $D_v = 100$

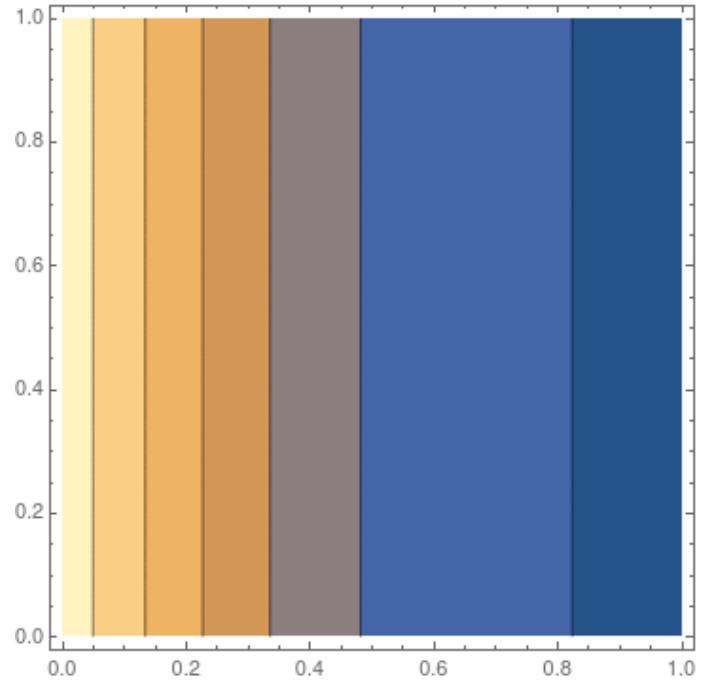


Рис. 14: Явная схема $v(t,x)$ $A=1$, $B=1$, $D_u = 1$, $D_v = 100$

3.2 Подготовительные расчеты

$$\begin{cases} \dot{u} = A + u^2v - (B + 1)u + D_u\Delta u, \\ \dot{v} = -u^2v + Bu + D_v\Delta v. \end{cases} \quad (19)$$

Границные условия:

$$\frac{\partial u}{\partial x}(0, t) = \frac{\partial u}{\partial x}(l, t) = \frac{\partial v}{\partial x}(0, t) = \frac{\partial v}{\partial x}(l, t) = 0. \quad (20)$$

Начальные условия:

$$u(x, 0) = A, \quad v(x, 0) = \frac{B}{A}. \quad (21)$$

$$u_1^{(k+1)} = u_0^{(k+1)} + \frac{\partial u}{\partial x}\Big|_0^{(k+1)} h + \frac{\partial^2 u}{\partial x^2}\Big|_0^{(k+1)} \frac{h^2}{2} \quad (22)$$

Выразим из первого уравнения брюсселятора 19 вторую производную по координате и подставим в предыдущее уравнение 22 (k - временная координата, j - пространственная):

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{D_u} \left(\frac{\partial u}{\partial t} - A - u^2v + (B + 1)u \right) \quad (23)$$

$$u_1^{(k+1)} = u_0^{(k+1)} + \frac{\partial u}{\partial x}\Big|_0^{(k+1)} h + \frac{h^2}{2D_u} \left(\frac{\partial u}{\partial t}\Big|_0^{(k+1)} - A - u^2v\Big|_0^{(k+1)} + (B + 1)u_0^{(k+1)} \right) \quad (24)$$

Выразим из 24 первую производную и приравняем ее нулю, так как граничным условием является нулевой поток.

$$\frac{\partial u}{\partial x} = \frac{1}{h} (u_1^{(k+1)} - u_0^{(k+1)} - \frac{h^2}{2D_u} (\frac{\partial u}{\partial t}\Big|_0^{(k+1)} - A - u^2v\Big|_0^{(k+1)} + (B + 1)u_0^{(k+1)})) = 0. \quad (25)$$

Разложим $(u^2v)\Big|_0^{(k+1)}$ по формуле Тейлора:

$$(u^2v)\Big|_0^{(k+1)} = (u^2)\Big|_0^{(k)} v_0^{(k+1)} + 2u_0^{(k)} v_0^{(k)} u_0^{(k+1)} - 2(u^2)\Big|_0^{(k)} v_0^{(k)}. \quad (26)$$

При этом

$$\left. \frac{\partial u}{\partial t} \right|_0^{(k+1)} = \frac{u_0^{(k+1)} - u_0^{(k)}}{\tau} \quad (27)$$

Подставим 26 и 27 в 25 и сгруппируем члены при $u_1^{(k+1)}$ и $u_0^{(k+1)}$.

$$b_0 u_0^{(k+1)} + c_0 u_1^{(k+1)} = d_0, \quad j = 0, \quad (28)$$

где

$$a_0 = 0, \quad b_0 = -\frac{1}{h} - \frac{h}{2D_u}((B+1) + \frac{1}{\tau} - 2u_0^{(k)}v_0^{(k)}), \quad c_0 = \frac{1}{h}, \quad d_0 = \frac{h^2}{2D_u}(-\frac{1}{\tau}u_0^{(k)} - A - (u^2)_0^{(k)}v_0^{(k+1)} + 2(u^2)_0^{(k)}v_0^{(k)}). \quad (29)$$

Аналогично поступим для второй границы.

$$u_{N-1}^{(k+1)} = u_N^{(k+1)} + \left. \frac{\partial u}{\partial x} \right|_N^{(k+1)} h + \left. \frac{\partial^2 u}{\partial x^2} \right|_N^{(k+1)} \frac{h^2}{2} \quad (30)$$

$$u_{N-1}^{(k+1)} = u_N^{(k+1)} + \left. \frac{\partial u}{\partial x} \right|_N^{(k+1)} h + \frac{h^2}{2D_u} \left(\left. \frac{\partial u}{\partial t} \right|_N^{(k+1)} - A - u^2 v \right|_N^{(k+1)} + (B+1)u_N^{(k+1)}) \quad (31)$$

$$\left. \frac{\partial u}{\partial x} \right|_N^{(k+1)} = \frac{1}{h}(u_{N-1}^{(k+1)} - u_N^{(k+1)} - \frac{h^2}{2D_u} \left(\left. \frac{\partial u}{\partial t} \right|_N^{(k+1)} - A - u^2 v \right|_N^{(k+1)} + (B+1)u_N^{(k+1)})) = 0. \quad (32)$$

$$\left. \frac{\partial u}{\partial t} \right|_N^{(k+1)} = \frac{u_N^{(k+1)} - u_N^{(k)}}{\tau} \quad (33)$$

$$a_N u_{N-1}^{(k+1)} + b_N u_N^{(k+1)} = d_N, \quad j = N, \quad (34)$$

где

$$a_N = \frac{1}{h}, \quad b_N = -\frac{1}{h} - \frac{h}{2D_u}((B+1) + \frac{1}{\tau} - 2u_N^{(k)}v_N^{(k)}), \quad c_N = 0, \quad d_N = \frac{h}{2D_u}(-\frac{1}{\tau}u_N^{(k)} - A - (u^2)_N^{(k)}v_N^{(k+1)} + 2(u^2)_N^{(k)}v_N^{(k)}). \quad (35)$$

Припишем к граничным конечно-разностным уравнениям 19 в виде:

$$\frac{u_j^{(k+1)} - u_j^{(k)}}{\tau} = \frac{D_u}{h^2} (u_{j+1}^{(k+1)} - 2u_j^{(k+1)} + u_{j-1}^{(k+1)}) + A + (u^2)_j^{(k)}v_j^{(k+1)} + 2u_j^{(k)}v_j^{(k)}u_j^{(k+1)} - 2(u^2)_j^{(k)}v_j^{(k)} - (B+1)u_j^{(k+1)}, \quad j = \overline{1, N-1}. \quad (36)$$

$$a_j u_{j-1}^{(k+1)} + b_j u_j^{(k+1)} + c_j u_{j+1}^{(k+1)} = d_j, \quad j = \overline{1, N-1}, \quad (37)$$

где

$$a_j = \frac{D_u}{h^2}, \quad b_j = -\frac{1}{\tau} - \frac{2D_u}{h^2} - (B+1) + 2u_j^{(k)}v_j^{(k)}, \quad c_j = \frac{D_u}{h^2}, \quad d_j = -(\frac{u_j^{(k)}}{\tau} + A + (u^2)_j^{(k)}v_j^{(k+1)} - 2(u^2)_j^{(k)}v_j^{(k)}). \quad (38)$$

По изложенному алгоритму поступим и со вторым уравнения брюсселятора.

$$b_0 v_0^{(k+1)} + c_0 v_1^{(k+1)} = d_0, \quad j = 0, \quad (39)$$

где

$$a_0 = 0, \quad b_0 = -\frac{1}{h} - \frac{h^2}{2D_v}(\frac{1}{\tau} + (u^2)_0^{(k)}), \quad c_0 = \frac{1}{h}, \quad d_0 = \frac{h^2}{2D_u}(-\frac{1}{\tau}v_0^{(k)} + 2u_0^{(k)}v_0^{(k)}u_0^{(k+1)} - 2(u^2)_0^{(k)}v_0^{(k)} - Bu_0^{(k+1)}). \quad (40)$$

$$a_N v_{N-1}^{(k+1)} + b_N v_N^{(k+1)} = d_N, \quad j = N, \quad (41)$$

где

$$a_0 = \frac{1}{h}, \quad b_0 = -\frac{1}{h} - \frac{h}{2D_v}(\frac{1}{\tau} + (u^2)_N^{(k)}), \quad c_N = 0, \quad d_N = \frac{h}{2D_v}(-\frac{1}{\tau}v_N^{(k)} + 2u_N^{(k)}v_N^{(k)}u_N^{(k+1)} - 2(u^2)_N^{(k)}v_N^{(k)} - Bu_N^{(k+1)}). \quad (42)$$

$$\frac{v_j^{(k+1)} - v_j^{(k)}}{\tau} = \frac{D_v}{h^2} (v_{j+1}^{(k+1)} - 2v_j^{(k+1)} + v_{j-1}^{(k+1)}) - (u^2)_j^{(k)} v_j^{(k+1)} - 2u_j^{(k)} v_j^{(k)} u_j^{(k+1)} + 2(u^2)_j^{(k)} v_j^{(k)} + Bu_j^{(k+1)}, j = \overline{1, N-1}. \quad (43)$$

$$a_j v_{j-1}^{(k+1)} + b_j v_j^{(k+1)} + c_j v_{j+1}^{(k+1)} = d_j, j = \overline{1, N-1}, \quad (44)$$

где

$$a_j = \frac{D_v}{h^2}, \quad b_j = -\frac{1}{\tau} - \frac{2D_u}{h^2} + (u^2)_j^{(k)}, \quad c_j = \frac{D_v}{h^2}, \quad d_j = -(\frac{v_j^{(k)}}{\tau} + 2(u^2)_j^{(k)} v_j^{(k)} + 2u_j^{(k)} v_j^{(k)} u_j^{(k+1)} + Bu_j^{(k+1)}). \quad (45)$$

Для каждого из уравнений получаем СЛАУ с трехдиагональной матрицей, решаемой методом прогонки:

$$A_j = -\frac{c_j}{b_j + a_j A_{j-1}}, \quad B_j = \frac{d_j - a_j B_{j-1}}{b_j + a_j A_{j-1}}, \quad A_0 = -\frac{c_0}{b_0}, \quad B_0 = \frac{d_0}{b_0}, \quad A_N = 0, j = \overline{0, N}; \quad (46)$$

$$u_j^{(k+1)} = A_j u_{j+1}^{(k+1)} + B_j, \quad u_N^{(k+1)} = B_N, j = N, N-1, \dots, 0. \quad (47)$$

3.3 Метод Кранка-Николсона

$$\begin{cases} \dot{u} = A + u^2 v - (B + 1)u + D_u \Delta u, \\ \dot{v} = -u^2 v + Bu + D_v \Delta v. \end{cases} \quad (48)$$

Границные условия:

$$\frac{\partial u}{\partial x}(0, t) = \frac{\partial u}{\partial x}(l, t) = \frac{\partial v}{\partial x}(0, t) = \frac{\partial v}{\partial x}(l, t) = 0. \quad (49)$$

Начальные условия:

$$u(x, 0) = A, \quad v(x, 0) = \frac{B}{A}. \quad (50)$$

Введем: р - пространственная координата, q - времененная.

Приблизим вторую производную в уравнениях разностной схемой Кранка-Николсона, перегруппируем члены и введем обозначения $r_u = \frac{D_u \tau}{2h^2}$, $r_v = \frac{D_v \tau}{2h^2}$, $f_u = f_u(u_p^{(q)}, v_p^{(q)}) = A + (u_p^{(q)})^2 v_p^{(q)} - (B + 1)u_p^{(q)}$, $f_v = f_v(u_p^{(q)}, v_p^{(q)}) = -(u_p^{(q)})^2 v_p^{(q)} - Bu_p^{(q)}$ (здесь τ - шаг времененной сетки, а h - пространственной):

$$u_p^{(q+1)} - u_p^{(q)} = \frac{D_u \tau}{2h^2} (u_{p+1}^{(q+1)} - 2u_p^{(q+1)} + u_{p-1}^{(q+1)}) + \frac{D_u \tau}{2h^2} (u_{p+1}^{(q)} - 2u_p^{(q)} + u_{p-1}^{(q)}) + \tau f_u = \\ r_u (u_{p+1}^{(q+1)} - 2u_p^{(q+1)} + u_{p-1}^{(q+1)}) + r_u (u_{p+1}^{(q)} - 2u_p^{(q)} + u_{p-1}^{(q)}) + \tau f_u. \quad (51)$$

$$u_p^{(q+1)} (1 + 2r_u) - r_u (u_{p+1}^{(q+1)} + u_{p-1}^{(q+1)}) = u_p^{(q)} + r_u (u_{p+1}^{(q)} - 2u_p^{(q)} + u_{p-1}^{(q)}) + \tau f_u. \quad (52)$$

$$\begin{cases} (1 + 2r_u)u_p^{q+1} - r_u (u_{p+1}^{q+1} + u_{p-1}^{q+1}) = u_p^{(q)} + r_u (u_{p+1}^{(q)} - 2u_p^{(q)} + u_{p-1}^{(q)}) + \tau f_u, \\ (1 + 2r_v)v_p^{q+1} - r_v (v_{p+1}^{q+1} + v_{p-1}^{q+1}) = v_p^{(q)} + r_v (v_{p+1}^{(q)} - 2v_p^{(q)} + v_{p-1}^{(q)}) + \tau f_v. \end{cases} \quad (53)$$

В левой части стоят три неизвестных величины, а в правой три известных. Получили тридиагональную матрицу. Учтем граничные условия:

$$u_0^{(q+1)} - u_0^{(q)} = r_u (u_1^{q+1} - 2u_0^{(q+1)} + u_{-1}^{(q+1)}) + r_u (u_1^{(q)} - 2u_0^{(q)} + u_{-1}^{(q)}) + \tau f_u. \quad (54)$$

Так как граничным условием является нулевой поток, $u_{-1}^{(q)} = u_1^{(q)}$ и уравнение перепишется в виде

$$u_0^{(q+1)} - u_0^{(q)} = 2r_u (u_1^{q+1} - u_0^{(q+1)}) + 2r_u (u_1^{(q)} - u_0^{(q)}) + \tau f_u. \quad (55)$$

$$(1 + 2r_u)u_0^{q+1} - 2r_u u_1^{q+1} = u_0^{(q)} + 2r_u (u_1^{(q)} - u_0^{(q)}) + \tau f_u. \quad (56)$$

Аналогично для второй границы и второй переменной.

Запишем в виде матрицы:

$$\begin{pmatrix} (1+2r_u) & -2r_u & 0 & 0 & 0 & \dots & 0 \\ -r_u & (1+2r_u) & -r_u & 0 & 0 & \dots & 0 \\ 0 & -r_u & (1+2r_u) & -r_u & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & -r_u & (1+2r_u) & -r_u \\ 0 & 0 & 0 & 0 & 0 & -2r_u & (1+2r_u) \end{pmatrix} \cdot \begin{pmatrix} u_0^{q+1} \\ u_1^{q+1} \\ u_2^{q+1} \\ \vdots \\ u_{n-1}^{q+1} \\ u_n^{q+1} \end{pmatrix} = \quad (57)$$

$$\begin{pmatrix} u_0^{(q)} + 2r_u(u_1^{(q)} - u_0^{(q)}) + \tau f_u \\ u_1^{(q)} + r_u(u_2^{(q)} - 2u_1^{(q)} + u_0^{(q)}) + \tau f_u \\ u_2^{(q)} + r_u(u_3^{(q)} - 2u_2^{(q)} + u_1^{(q)}) + \tau f_u \\ \vdots \\ u_{n-1}^{(q)} + r_u(u_n^{(q)} - 2u_{n-1}^{(q)} + u_{n-2}^{(q)}) + \tau f_u \\ u_n^{(q)} + 2r_u(u_{n-1}^{(q)} - u_n^{(q)}) + \tau f_u \end{pmatrix} \quad (58)$$

Аналогичные выражения можно получить и для v .

$$\begin{pmatrix} (1+2r_v) & -2r_v & 0 & 0 & 0 & \dots & 0 \\ -r_v & (1+2r_v) & -r_v & 0 & 0 & \dots & 0 \\ 0 & -r_v & (1+2r_v) & -r_v & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & -r_v & (1+2r_v) & -r_v \\ 0 & 0 & 0 & 0 & 0 & -2r_v & (1+2r_v) \end{pmatrix} \cdot \begin{pmatrix} v_0^{q+1} \\ v_1^{q+1} \\ v_2^{q+1} \\ \vdots \\ v_{n-1}^{q+1} \\ v_n^{q+1} \end{pmatrix} = \quad (59)$$

$$\begin{pmatrix} v_0^{(q)} + 2r_v(v_1^{(q)} - v_0^{(q)}) + \tau f_v \\ v_1^{(q)} + r_v(v_2^{(q)} - 2v_1^{(1)} + v_0^{(q)}) + \tau f_v \\ v_2^{(q)} + r_v(v_3^{(q)} - 2v_2^{(1)} + v_1^{(q)}) + \tau f_v \\ \vdots \\ v_{n-1}^{(q)} + r_v(v_n^{(q)} - 2v_{n-1}^{(1)} + v_{n-2}^{(q)}) + \tau f_v \\ v_n^{(q)} + 2r_v(v_{n-1}^{(q)} - v_n^{(q)}) + \tau f_v \end{pmatrix} \quad (60)$$

Так как обе матрицы имеют диагональное преобладание, то их можно решать методом прогонки.

3.4 Границные условия

Первый способ записать граничные условия:

$$\dot{u} = A + u^2 v - (B + 1)u + D_u \Delta u. \quad (61)$$

Так как на границе нулевой поток, то

$$u_1^{(k)} = u_0^{(k)} + \frac{\partial u}{\partial x} \Big|_0^{(k)} h + \frac{\partial^2 u}{\partial x^2} \Big|_0^{(k)} \frac{h^2}{2} = u_0^{(k)} + \frac{\partial^2 u}{\partial x^2} \Big|_0^{(k)} \frac{h^2}{2}. \quad (62)$$

Подставляем вторую производную из 62 в 61.

$$\dot{u} = A + u^2 v - (B + 1)u + \frac{2Du}{h^2} (u_1^{(k)} - u_0^{(k)}). \quad (63)$$

Производная по времени

$$u_0^{(k+1)} = u_0^{(k)} + \tau \dot{u} \Big|_0^k. \quad (64)$$

Значит,

$$u_0^{(k+1)} = u_0^{(k)} + \tau(A + (u_0^{(k)})^2 v_0^{(k)} - (B + 1)u_0^{(k)} + \frac{2Du}{h^2}(u_1^{(k)} - u_0^{(k)})). \quad (65)$$

Аналогично:

$$u_{N-1}^{(k+1)} = u_{N-1}^{(k)} + \tau(A + (u_{N-1}^{(k)})^2 v_{N-1}^{(k)} - (B + 1)u_{N-1}^{(k)} + \frac{2Du}{h^2}(u_{N-2}^{(k)} - u_{N-1}^{(k)})). \quad (66)$$

$$v_0^{(k+1)} = v_0^{(k)} + \tau(-(u_0^{(k)})^2 v_0^{(k)} + Bu_0^{(k)} + \frac{2Du}{h^2}(v_1^{(k)} - v_0^{(k)})). \quad (67)$$

$$v_{N-1}^{(k+1)} = v_{N-1}^{(k)} + \tau(-(u_{N-1}^{(k)})^2 v_{N-1}^{(k)} + Bu_{N-1}^{(k)} + \frac{2Du}{h^2}(v_{N-2}^{(k)} - v_{N-1}^{(k)})). \quad (68)$$

Второй более простой способ записать граничные условия:

$$\frac{\partial u}{\partial x} \Big|_{x=0}^q = 0 = \frac{u_1^q - u_0^q}{h}. \quad (69)$$

Отсюда

$$u_1^q = u_0^q, \quad u_n^q = u_{n-1}^q, \quad v_1^q = v_0^q, \quad v_n^q = v_{n-1}^q. \quad (70)$$

Третий способ записи граничных условий - метод фиктивного узла - реализован в предыдущем пункте.

3.5 Исследование устойчивости разностной схемы

3.5.1 Признак максимума

Рассмотрим первое уравнение брюсселятора:

$$\frac{u_p^{(q+1)} - u_p^{(q)}}{\tau} = D(u_p^{(q+1)} + u_p^{(q)}) + f_u. \quad (71)$$

Здесь D - оператор второй производной, равный $Du_p = \frac{u_{p+1} - 2u_p + u_{p-1}}{2h^2}$. Так же это уравнение можно записать

$$\sum_{Q \in O(P)} M_h(P, Q)u(Q) = f_u(P). \quad (72)$$

Мы рассматриваем сетку с шагом h окрестности точки P . Приведем к каноническому виду:

$$A(p)u(P) = \sum_{Q \in O(P) \setminus P} B(P, Q)u(Q) + f_u(P). \quad (73)$$

В качестве центрального узла шаблона P выберем точку $P(x_p, t_{q+1})$, тогда

$$O(P) \setminus P = \{Q_1(x_p, t_q), Q_2(x_{p-1}, t_q), Q_3(x_{p+1}, t_q), Q_4(x_{p-1}, t_{q+1}), Q_5(x_{p+1}, t_{q+1})\}. \quad (74)$$

Из предыдущих вычислений:

$$(1 + 2r_u)u_p^{q+1} = r_u(u_{p+1}^{q+1} + u_{p-1}^{q+1}) + r_u(u_{p+1}^{(q)} + u_{p-1}^{(q)}) + (1 - 2r_u)u_p^{(q)} + \tau f_u. \quad (75)$$

Тогда коэффициенты канонической формы разностного уравнения будут иметь вид:

$$A(P) = 1 + 2r_u, \quad B(P, Q_1) = (1 - 2r_u), \quad B(P, Q_2) = B(P, Q_3) = B(P, Q_4) = B(P, Q_5) = r_u. \quad (76)$$

Воспользуемся принципом максимума. Для устойчивости разностной схемы достаточно, чтобы $A(P) > 0$, $B(P, Q_m) \geq 0 \quad \forall m$ и $D(P) = A(P) - \sum_{Q \in O(P) \setminus P} B(P, Q) \geq 0$. В нашем случае эти условия выполняются: $A(P) = 1 + 2r_u > 0$

всегда, $B(P, Q_1) = (1 - 2r_u) \geq 0$, если $r_u < \frac{1}{2}$, $B(P, Q_2) = B(P, Q_3) = B(P, Q_4) = B(P, Q_5) = r_u \geq 0$ всегда и $D(P) = 1 + 2r_u - (1 - 2r_u) - 4r_u = 0$ всегда. Аналогичные условия можно написать для второго уравнения брюсселятора. Однако это условие является достаточным, но не необходимым. Его отсутствие неизбежно влечет к потери устойчивости.

3.5.2 Энергетический критерий устойчивости

Воспользуемся энергетическим критерием устойчивости. Рассмотрим первое уравнение брюсселятора:

$$\frac{u_p^{(q+1)} - u_p^{(q)}}{\tau} = \frac{D_u}{2} \Lambda_{xx}(u_p^{(q+1)} + u_p^{(q)}) + f_u. \quad (77)$$

Здесь Λ_{xx} - оператор второй производной, равный $\Lambda_{xx}u_p = \frac{u_{p+1}-2u_p+u_{p-1}}{h^2}$. Приведем к каноническому виду:

$$(E - \frac{D_u \tau}{2} \Lambda_{xx}) \frac{u_p^{(q+1)} - u_p^{(q)}}{\tau} - D_u \Lambda_{xx} u_p^{(q)} = f_u. \quad (78)$$

Необходимым и достаточным условием устойчивости будет $E - \frac{D_u \tau}{2} \Lambda_{xx} \geq -\frac{\tau D_u}{2} \Lambda_{xx}$ или $E \geq 0$, что верно при любых шагах по пространству и времени. Аналогично для второго уравнения брюсселятора. То есть наша схема Кранка-Николсона абсолютно устойчива.

3.6 Исследование на устойчивость распределенного случая

Рассмотрим более общий случай:

$$\begin{cases} \frac{\partial u}{\partial t} = P(u, v, x) + D_u \frac{\partial^2 u}{\partial x^2}, \\ \frac{\partial v}{\partial t} = Q(u, v, x) + D_v \frac{\partial^2 v}{\partial x^2}. \end{cases} \quad (79)$$

Введем ξ и η - малые отклонения от решений системы $P(u, v) = 0$, $Q(u, v) = 0$. Тогда уравнения перепишутся:

$$\begin{cases} \frac{\partial \xi}{\partial t} = a\xi + b\eta + D_u \frac{\partial^2 \xi}{\partial x^2}, \\ \frac{\partial \eta}{\partial t} = c\xi + d\eta + D_v \frac{\partial^2 \eta}{\partial x^2}, \end{cases} \quad (80)$$

где

$$a = \frac{\partial P}{\partial u}, \quad b = \frac{\partial P}{\partial v}, \quad c = \frac{\partial Q}{\partial u}, \quad d = \frac{\partial Q}{\partial v}. \quad (81)$$

Решение будем искать в виде $\xi(t, x) = \bar{A}e^{pt}e^{ikx}$, $\eta(t, x) = \bar{B}e^{pt}e^{ikx}$.

Подставим в систему и сократим на экспоненты, получим:

$$\begin{cases} \bar{A}p = a\bar{A} + b\bar{B} - D_u k^2 \bar{A}, \\ \bar{B}p = c\bar{A} + d\bar{B} - D_v k^2 \bar{B}. \end{cases} \quad (82)$$

\bar{A} и \bar{B} не равны нулю, если

$$(p - a + k^2 D_u)(p - d + k^2 D_v) - bc = 0. \quad (83)$$

$$p^2 + p((D_u + D_v)k^2 - a - d) + (a - k^2 D_u)(d + k^2 D_v) + bc = 0. \quad (84)$$

Решения этого уравнения:

$$p_{1,2} = \frac{a + d - (D_u + D_v)k^2 \pm \sqrt{(a + d - k^2(D_u + D_v))^2 - 4(a - k^2 D_u)(d + k^2 D_v) - 4bc}}{2}. \quad (85)$$

Или, вводя обозначения $\Delta = (D_u + D_v)k^2 - a - d$, $\Sigma = (a - k^2 D_u)(d + k^2 D_v) + bc$,

$$p^2 + \Delta p + \Sigma = 0. \quad (86)$$

$$p_{1,2} = \frac{-\Delta \pm \sqrt{\Delta^2 - 4\Sigma}}{2}. \quad (87)$$

Знак действительной части $p_{1,2}$ показывает, будет ли решением устойчивым.

В нашем случае

$$a = \frac{\partial P}{\partial u} = B - 1, \quad b = \frac{\partial P}{\partial v} = A^2, \quad c = \frac{\partial Q}{\partial u} = -B, \quad d = \frac{\partial Q}{\partial v} = -A^2. \quad (88)$$

и дисперсионное уравнение имеет вид:

$$(p - B + 1 + k^2 D_u)(p + A^2 + k^2 D_v) + BA^2 = 0. \quad (89)$$

$$p_{1,2} = \frac{B - 1 - A^2 - (D_u + D_v)k^2 \pm \sqrt{(B - 1 - A^2 - k^2(D_u + D_v))^2 - 4A^2B + 4(B - 1 - k^2D_u)(A^2 + k^2D_v)}}{2}. \quad (90)$$

Построим график $p_{1,2}(k)$:

```

Assuming[{k \[Element] Reals, k > 0},
Block[{B = 3, A = 1, Du = 1, Dv = 100},
p1[k_] := 0.5*(B - 1 - A^2 - (Du + Dv) k^2 + Sqrt[(B - 1 - A^2 - (Du + Dv) k^2)^2 - 4 A^2 B + 4 (B - 1 - k^2 D_u) (A^2 + k^2 D_v)]];
p2[k_] := 0.5*(B - 1 - A^2 - (Du + Dv) k^2 - Sqrt[(B - 1 - A^2 - (Du + Dv) k^2)^2 - 4 A^2 B + 4 (B - 1 - k^2 D_u) (A^2 + k^2 D_v)]);
realp1[k_] := Re@p1[k];
realp2[k_] := Re@p2[k];
Show[Plot[{realp1[k], realp2[k]}, {k, 0, 2}, PlotRange > {{0, 2}, {2, 2}}]]
]
}

```

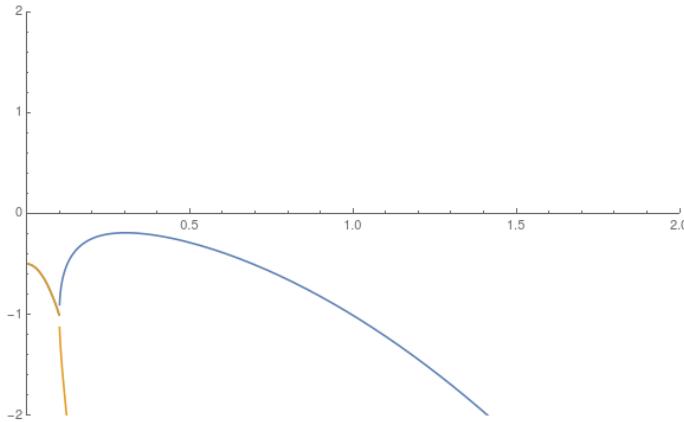


Рис. 15: Дисперсионная кривая $A=1$, $B=1$, $D_u = 1$, $D_v = 100$

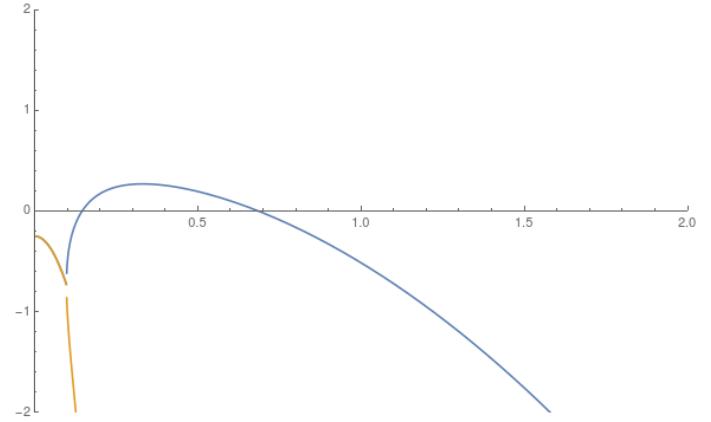


Рис. 16: Дисперсионная кривая $A=1$, $B=1.5$, $D_u = 1$, $D_v = 100$

3.7 Неустойчивость Тьюринга

Определим условия возникновения диффузионной неустойчивости (обусловленной только процессами диффузии). Это означает, что однородное стационарное состояние будет устойчивым по отношению к малым однородным возмущениям (соответствующая точечная система устойчива) и не устойчива по отношению к малым пространственно неоднородным возмущениям.

В нераспределенном случае дисперсионное уравнение 16 выглядит так:

$$\lambda^2 + \delta\lambda + \sigma = 0, \quad (91)$$

где $\delta = -a - d$ и $\sigma = ad - bc$. Решение устойчиво, если $\delta > 0$ и $\sigma > 0$.

Распределенный случай 92:

$$p^2 + \Delta p + \Sigma = 0, \quad (92)$$

где $\Delta = (D_u + D_v)k^2 - a - d$, $\Sigma = k^4 D_u D_v - k^2(a D_v + d D_u) + ad - bc$. Решение неустойчиво, если либо $\Delta < 0$, либо $\Sigma < 0$. Так как $\delta > 0$, то $\Delta = (D_u + D_v)k^2 + \delta$ так же будет больше нуля. Значит, для наблюдения неустойчивости, необходимо, чтобы $\Sigma < 0$.

Таким образом для возникновения неустойчивости Тьюринга необходимо, чтобы

$$\begin{cases} a + d < 0, \\ ad - bc > 0, \\ k^4 D_u D_v - k^2(aD_v + dD_u) + ad - bc < 0. \end{cases} \quad (93)$$

В случае брюсселятора

$$\begin{cases} B - 1 - A^2 < 0, \\ A^2 > 0, \\ k^4 D_u D_v - k^2(aD_v + dD_u) + A^2 < 0. \end{cases} \quad (94)$$

Рассмотрим третье условие. Так как $k^4 D_u D_v$ и $ad - bc$ больше нуля, то Σ может стать меньше нуля только засчет $-k^2(aD_v + dD_u)$. То есть для неустойчивости необходимо, чтобы $aD_v + dD_u > 0$. Получили систему:

$$\begin{cases} a + d < 0, \\ aD_v + dD_u > 0. \end{cases} \quad (95)$$

Чтобы она выполнялась, необходимо, чтобы a и d были разных знаков. В случае брюсселятора $d = -A^2 < 0$, то есть $a = B - 1 > 0$. То есть $B > 1$.

Чтобы было $a + d < 0$, нужно, чтобы $|a| < |d|$. Иными словами $\frac{B-1}{A^2} < 1$.

$$aD_v + dD_u > 0 \quad (96)$$

$$D_u(a\frac{D_v}{D_u} + d) > 0 \quad (97)$$

Другими словами необходимо, чтобы $\frac{D_v}{D_u} > 1$, $D_v > D_u$. Разница коэффициентов диффузии является необходимым условием наличия неустойчивости Тьюринга.

Теперь проанализируем $f(k^2) = k^4 D_u D_v - k^2(aD_v + dD_u) + ad - bc$. Дискриминант $D = (aD_v + dD_u)^2 - 4D_u D_v + \sigma$. Чтобы парабола $f(k^2)$ пересекала ось абсцисс, нужно, чтобы $D \geq 0$. Критическое состояние $D = 0$, решим его относительно $D_{cr} = \frac{D_v}{D_u}$:

$$(aD_v + dD_u)^2 - 4D_u D_v + \sigma = 0, \quad (98)$$

$$(aD_{cr} + d)^2 - 4D_{cr}\sigma = 0, \quad (99)$$

$$a^2 D_{cr}^2 + 2(da - 2\sigma)D_{cr} + d^2 = 0. \quad (100)$$

Его решения:

$$D_{cr1,2} = \frac{2\sigma - ad \pm 2\sqrt{\sigma}\sqrt{\sigma - ad}}{a^2}. \quad (101)$$

Так как отношение концентраций должно быть числом положительным, то подходит только корень с плюсом. Делаем вывод, что для наблюдения диффузной неустойчивости необходимо, чтобы

- $a + d < 0$, $a > 0, d < 0$
- $ad - bc > 0$
- $\frac{D_v}{D_u} > \frac{2\sigma - ad + 2\sqrt{\sigma}\sqrt{\sigma - ad}}{a^2}$

При этом

$$k_{cr}^2 = \frac{aD_{cr} + d}{2D_u D_{cr}}. \quad (102)$$

А число диссипативных структур будет равно $[\frac{k_{cr}}{2\pi}]$.

В случае брюсселятора:

$$D_{cr} = \frac{A^2}{(\sqrt{B} - 1)^2}, \quad k_{cr}^2 = \frac{\sqrt{B} - 1}{D_u}, \quad n = \frac{k_{cr}}{2\pi} = \frac{1}{2\pi} \sqrt{\frac{\sqrt{B} - 1}{D_u}} \quad (103)$$

3.8 Реализация на С

Напишем программу, подсчитывающую значения концентраций, используя метод разделения по физическим процессам. Разностную схему приблизим шеститочечной схемой Кранка-Николсона.

Опишем алгоритм подбора параметров пространственно-временной сетки:

1. Зафиксируем параметры брюсселятора: коэффициенты A, B, D_u и D_v .
2. Строим график зависимости $Re(p_{1,2}(k))$ для данных параметров.
3. Находим k_{min} и k_{max} , соответствующие смене знака реальной части $p_{1,2}$ и, соответственно, наступлению зоны седловой неустойчивости.
4. $\lambda_{min} = \frac{2\pi}{k_{max}}, \lambda_{max} = \frac{2\pi}{k_{min}}$ - границы интерала длин волн, которые будут неустойчивы.
5. λ_{max} должна войти в пространственную область несколько раз, поэтому $b \approx 5\lambda_{max}$.
6. В волну должно укладываться несколько шагов сетки, тогда $h \approx \frac{\lambda_{min}}{10}$.
7. Всего шагов по пространству будет $\frac{b}{h} \approx 50 \frac{\lambda_{max}}{\lambda_{min}} \approx 100$.
8. Исходя из h подберем шаг временной сетки так, чтобы $\frac{D_v \tau}{h^2} < \frac{1}{2}$, то есть $\tau < \frac{h^2}{2D_v}$.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 #define PREF "brus"
6 #define SUFF ".dat"
7
8 const float A = 1;
9 const float B = 1.5;
10 const float Du = 1;
11 const float Dv = 100;
12
13 float F1( float u, float v)
14 {
15     return (A + u*u*v * (B+1)*u);
16 }
17
18 float F2( float u, float v)
19 {
20     return ( u*u*v + B*u );
21 }
22
23
24 float runge_kutt( float h, float u, float v, int f)
25 {
26     float k11 = 0, k12 = 0, k13 = 0, k14 = 0;
27     float k21 = 0, k22 = 0, k23 = 0, k24 = 0;
28     k11 = h*F1(u, v);
29     k21 = h*F2(u, v);
30     k12 = h*F1(u + k11/2, v + k21/2);
31     k22 = h*F2(u + k11/2, v + k21/2);
32     k13 = h*F1(u + k12/2, v + k22/2);
33     k23 = h*F2(u + k12/2, v + k22/2);
34     k14 = h*F1(u + k13/2, v + k23/2);
35     k24 = h*F2(u + k13/2, v + k23/2);
36     u = u + (k11 + 2*k12 + 2*k13 + k14)/6;
37     v = v + (k21 + 2*k22 + 2*k23 + k24)/6;
38
39     if(f==1) return u;
40     if(f==2) return v;
41 }
42
43 void solve_tridiagonal( float * x, int X, float * a, float * b, float * c, float* d) {

```

```

44 /*
45 solves Ax = d where A is a tridiagonal matrix consisting of vectors a, b, c
46 x contains the input vector x indexed from 0 to X - 1 inclusive
47 X number of equations (length of vector x)
48 a subdiagonal (means it is the diagonal below the main diagonal), indexed from 1 to X - 1
49 inclusive
50 b the main diagonal, indexed from 0 to X - 1 inclusive
51 c superdiagonal (means it is the diagonal above the main diagonal), indexed from 0 to X - 2
52 inclusive
53 d right part of equation, indexed from 0 to X - 1
54 */
55
56 float q[X];
57 float p[X];
58
59 p[0] = c[0] / b[0];
60 q[0] = d[0] / b[0];
61
62 for (int ix = 1; ix <= X; ix++) {
63     float m = 1.0 / (b[ix - 1] - a[ix - 1] * p[ix - 1]);
64     p[ix] = c[ix - 1] * m;
65     q[ix] = (d[ix - 1] + a[ix - 1] * q[ix - 1]) * m;
66 }
67 x[X - 1] = q[X];
68
69 for (int ix = X - 2; ix >= 0; ix)
70     x[ix] = q[ix + 1] + p[ix + 1] * x[ix + 1];
71 }
72
73 int main() {
74     float a = 0;
75     float b = 230;
76     float T = 1000;
77     float t0 = 0;
78     float h = 0.9;
79     float tau = 0.001;
80     int Nx = (b - a) / h;
81     int Nt = (T - t0) / tau;
82     int count = 0;
83
84     float ru = Du*tau/2/h/h;
85     float rv = Dv*tau/2/h/h;
86
87     float au[Nx+1]; float bu[Nx+1]; float cu[Nx+1];
88     float av[Nx+1]; float bv[Nx+1]; float cv[Nx+1];
89     au[0] = 0.; av[0] = 0.;
90     au[Nx] = 2*ru;
91     av[Nx] = 2*rv;
92
93     for (int i=1; i<Nx; i++){
94         av[i] = rv;
95         au[i] = ru;
96     }
97     for (int i=0; i<=Nx; i++){
98         bu[i] = 1+2*ru;
99         bv[i] = 1+2*rv;
100    }
101
102    cu[Nx] = 0.;
103    cv[Nx] = 0.;
104    cu[0] = 2*ru;
105    cv[0] = 2*rv;
106    for (int i=1; i<Nx; i++){
107        cv[i] = rv;
108        cu[i] = ru;
109    }

```

```

111 float fu [ 2 ][ Nx+1];
112 float fv [ 2 ][ Nx+1];
113
114 for( int j = 0; j < 2; j++){
115     for( int i = 0; i <= Nx; i++){
116         fu [ j ][ i ] = 0.;
117         fv [ j ][ i ] = 0.;
118     }
119 }
120     float noiseu = 0.005*(rand()%100);
121     float noisev = 0.005*(rand()%100);
122 for( int i = 0; i <= Nx; i++){
123     fv [ 0 ][ i ] = B/A + noisev;
124     fu [ 0 ][ i ] = A + noiseu;
125 }
126
127 float urightpart [Nx+1];
128 float vrightpart [Nx+1];
129 float u[Nx+1];
130 float v[Nx+1];
131
132 for( int i = 0; i <= Nx; i++){
133     v[ i ] = 0;
134     u[ i ] = 0;
135 }
136 int number_of_files = Nt+1;
137
138
139 char buf[BUFSIZ];
140 FILE* fp ;
141
142 while( count*tau<T){
143
144     for( int i = 0; i<=Nx; i++){
145         fu [ 1 ][ i ] = runge_kutt(tau , fu [ 0 ][ i ], fv [ 0 ][ i ], 1);
146         fv [ 1 ][ i ] = runge_kutt(tau , fu [ 0 ][ i ], fv [ 0 ][ i ], 2);
147     }
148
149     urightpart [ 0 ] = fu [ 1 ][ 0 ] + 2*ru*(fu [ 0 ][ 1 ] - fu [ 0 ][ 0 ]) ;
150     vrightpart [ 0 ] = fv [ 1 ][ 0 ] + 2*rv*(fv [ 0 ][ 1 ] - fv [ 0 ][ 0 ]) ;
151     urightpart [ Nx ] = fu [ 1 ][ Nx ] + 2*ru*(fu [ 0 ][ Nx-1 ] - fu [ 0 ][ Nx ]) ;
152     vrightpart [ Nx ] = fv [ 1 ][ Nx ] + 2*rv*(fv [ 0 ][ Nx-1 ] - fv [ 0 ][ Nx ]) ;
153
154     for( int i = 1; i < Nx; i++){
155         urightpart [ i ] = fu [ 1 ][ i ]+ru*(fu [ 0 ][ i+1 ] - 2*fu [ 0 ][ i ] + fu [ 0 ][ i-1 ]) ;
156         vrightpart [ i ] = fv [ 1 ][ i ]+rv*(fv [ 0 ][ i+1 ] - 2*fv [ 0 ][ i ] + fv [ 0 ][ i-1 ]) ;
157     }
158
159 solve_tridiagonal(u, Nx+1, au, bu, cu, urightpart);
160 solve_tridiagonal(v, Nx+1, av, bv, cv, vrightpart);
161 for( int i = 0; i < Nx+1; i++){
162     fu [ 1 ][ i ] = u[ i ];
163     fv [ 1 ][ i ] = v[ i ];
164 }
165 count++;
166
167     sprintf(buf, "%s%d%s", PREF, (count 1), SUFF);
168 if(count%1000==0){
169     fp = fopen(buf, "w");
170     for( int i = 0; i <= Nx; i++){
171         fprintf(fp, "%f %f %f %f \n", (count 1)*tau, i*h, fu [ 1 ][ i ], fv [ 1 ][ i ]);
172     }
173     fclose(fp);
174 }
175     for( int i = 0; i <= Nx; i++){
176         fu [ 0 ][ i ] = fu [ 1 ][ i ];
177         fv [ 0 ][ i ] = fv [ 1 ][ i ];
178     }

```

```

180 for( int i= 0; i <BUFSIZ; i++){
181     buf[ i ] = 0;
182 }
183 }
184
185 return 0;
186 }
```

Результатом исполнения кода является пакет данных, на каждый временной помежуток свой файл. Далее в gnuplot строятся графики зависимостей концентраций от координаты и объединяются в gif файл.

Приведем исполняемый скрипт:

```

set term gif animate
set output "animate.gif"
set yr [0.0:8.0]
do for [t=999:999999:1000] {
    outfile = sprintf('data%d.png', t)
    infile = sprintf('brus%d.dat', t)
    plot infile u 2:3 w l lt rgb "#0480ad", infile u 2:4 w l lt rgb "#7c00ad"
}
set output
```

Приведем примеры слайдов:

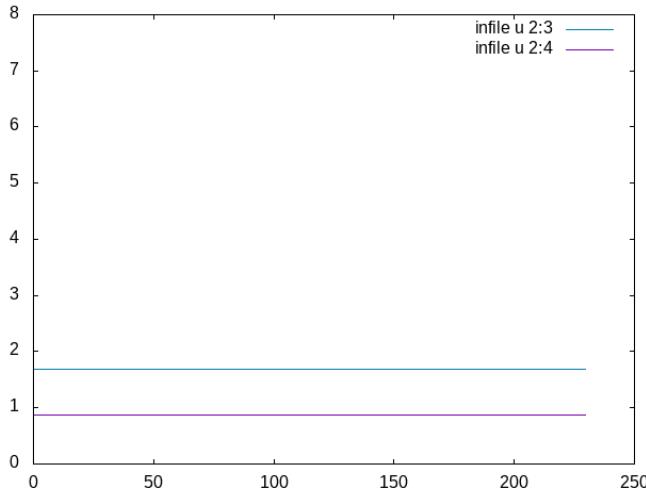


Рис. 17: $A = 1$, $B=1.5$, $D_u = 1$, $D_v = 100$, $T=999$, график зависимости концентраций от x

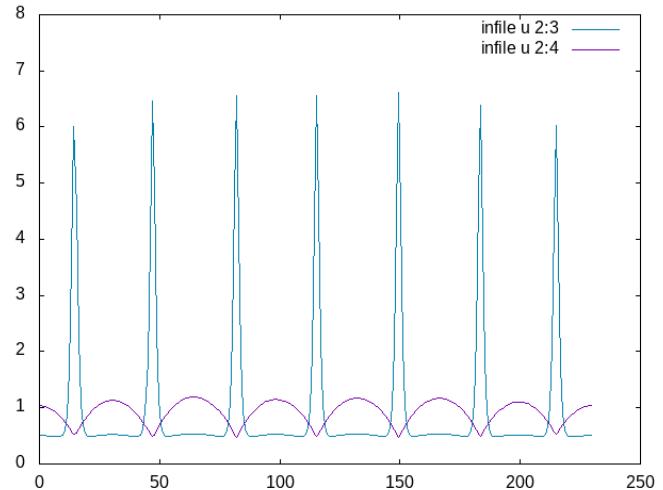


Рис. 18: $A = 1$, $B=1.5$, $D_u = 1$, $D_v = 100$, $T=999999$, график зависимости концентраций от x

Также можно построить 3D графики зависимости концентраций от x, t :

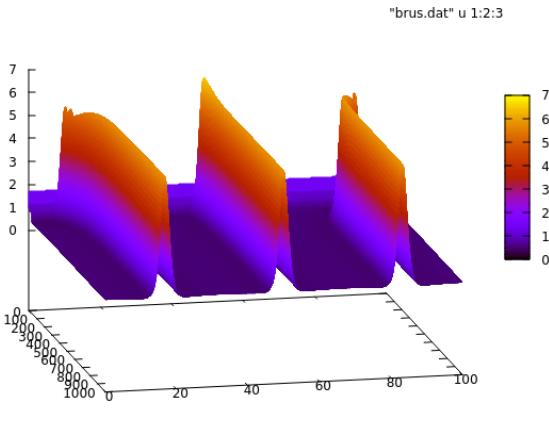


Рис. 19: A =1, B=1.5, Du = 1, Dv = 100, $u(t, x)$

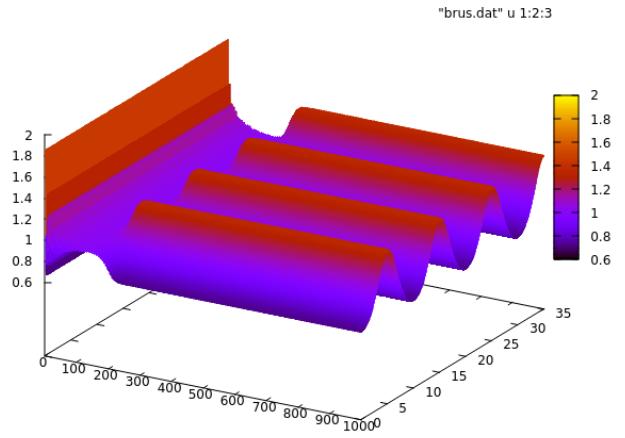


Рис. 20: A =1, B=1.9, Du = 1, Dv = 7, $u(t, x)$

3.9 Распределенный случай 2D

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 #define PREF "brus2da45b75"
6 #define SUFF ".dat"
7
8
9 const float A = 4.5;
10 const float B = 7.5;
11 const float Dux = 2;
12 const float Dvx = 16;
13 const float Duy = 2;
14 const float Dvy = 16;
15
16 float F1( float u, float v)
17 {
18     return (A + u*u*v * (B+1)*u);
19 }
20
21 float F2( float u, float v)
22 {
23     return ( u*u*v + B*u );
24 }
25
26 float runge_kutt( float h, float u, float v, int f)
27 {
28     float k11 = 0, k12 = 0, k13 = 0, k14 = 0;
29     float k21 = 0, k22 = 0, k23 = 0, k24 = 0;
30     k11 = h*F1(u, v);
31     k21 = h*F2(u, v);
32     k12 = h*F1(u + k11/2, v + k21/2);
33     k22 = h*F2(u + k11/2, v + k21/2);
34     k13 = h*F1(u + k12/2, v + k22/2);
35     k23 = h*F2(u + k12/2, v + k22/2);
36     k14 = h*F1(u + k13/2, v + k23/2);
37     k24 = h*F2(u + k13/2, v + k23/2);
38     u = u + (k11 + 2*k12 + 2*k13 + k14)/6;
39     v = v + (k21 + 2*k22 + 2*k23 + k24)/6;
40
41     if(f==1) return u;
42     if(f==2) return v;
43 }
44

```

```

45
46 void solve_tridiagonal(float * x, int X, float * a, float * b, float * c, float* d) {
47 /*
48 solves Ax = d where A is a tridiagonal matrix consisting of vectors a, b, c
49 x contains the input vector x indexed from 0 to X - 1 inclusive
50 X number of equations (length of vector x)
51 a subdiagonal (means it is the diagonal below the main diagonal), indexed from 1 to X - 1
52 inclusive
53 b the main diagonal, indexed from 0 to X - 1 inclusive
54 c superdiagonal (means it is the diagonal above the main diagonal), indexed from 0 to X - 2
55 inclusive
56 d right part of equation, indexed from 0 to X - 1
57 */
58
59 float q[X];
60 float p[X];
61
62 p[0] = c[0] / b[0];
63 q[0] = d[0] / b[0];
64
65 for (int ix = 1; ix <= X; ix++) {
66     float m = 1.0 / (b[ix - 1] - a[ix - 1] * p[ix - 1]);
67     p[ix] = c[ix - 1] * m;
68     q[ix] = (d[ix - 1] + a[ix - 1] * q[ix - 1]) * m;
69 }
70 x[X - 1] = q[X];
71
72 for (int ix = X - 2; ix >= 0; ix)
73     x[ix] = q[ix + 1] + p[ix + 1] * x[ix + 1];
74 }
75
76 int main() {
77
78 float x0 = 0;
79 float x1 = 30;
80 float y0 = 0;
81 float y1 = 30;
82 float T = 10;
83 float t0 = 0;
84 float h = 0.1;
85 float tau = 0.001;
86 int Ny = (y1 - y0)/h;
87 int Nx = (x1 - x0) / h;
88 int Nt = (T - t0) / tau;
89
90 int count = 0;
91
92 float rux = Dux*tau/2/h/h;
93 float rvx = Dvx*tau/2/h/h;
94 float ruy = Duy*tau/2/h/h;
95 float rvy = Dvy*tau/2/h/h;
96
97 float aux[Nx+1]; float bux[Nx+1]; float cux[Nx+1];
98 float avx[Nx+1]; float bvx[Nx+1]; float cvx[Nx+1];
99 float auy[Ny+1]; float buy[Ny+1]; float cuy[Ny+1];
100 float avy[Ny+1]; float bvy[Ny+1]; float cvy[Ny+1];
101
102 aux[0] = 0.; avx[0] = 0.;
103 aux[Nx] = 2*rux;
104 avx[Nx] = 2*rvx;
105 auy[0] = 0.; avy[0] = 0.;
106 auy[Ny] = 2*ruy;
107 avy[Ny] = 2*rvy;
108
109 for (int i=1; i<Nx; i++){
110     avx[i] = rvx;
111     aux[i] = rux;
112 }
113 for (int i=1; i<Ny; i++){

```

```

112     avy[ i ] = rvy;
113     auy[ i ] = ruy;
114 }
115 for( int i=0; i<=Nx; i++){
116     bux[ i ] = 1+2*rux;
117     bvx[ i ] = 1+2*rvx;
118 }
119 for( int i=0; i<=Ny; i++){
120     buy[ i ] = 1+2*ruy;
121     bvy[ i ] = 1+2*rvy;
122 }
123 cux[Nx] = 0.; cux[ 0 ] = 2 * rux;
124 cvx[Nx] = 0.; cvx[ 0 ] = 2 * rvx;
125 cuy[Ny] = 0.; cuy[ 0 ] = 2 * ruy;
126 cvy[Ny] = 0.; cvy[ 0 ] = 2 * rvy;
127 for( int i=1; i<Nx; i++){
128     cvx[ i ] = rvx;
129     cux[ i ] = rux;
130 }
131 for( int i=1; i<Ny; i++){
132     cvy[ i ] = rvy;
133     cuy[ i ] = ruy;
134 }
135
136 float fu [ 4 ][ Nx+1 ][ Ny+1 ];
137 float fv [ 4 ][ Nx+1 ][ Ny+1 ];
138
139 for( int j = 0; j < 4; j++){
140     for( int k = 0; k <= Ny; k++){
141         for( int i = 0; i <= Nx; i++){
142             fu [ j ][ i ][ k ] = 0.;
143             fv [ j ][ i ][ k ] = 0.;

144         }
145     }
146 }
147 /* float noiseu = 0.005*(rand()%100);
148    float noisev = 0.005*(rand()%100); */
149 for( int i = 0; i <= Nx; i++){
150     for( int k = 0; k <= Ny; k++){
151         fv [ 0 ][ i ][ k ] = B/A + 0.0001*(rand()%100); //+ noiseu;
152         fu [ 0 ][ i ][ k ] = A + 0.0001*(rand()%100); //+ noisev;
153     }
154
155 // MAX(NX,NY)+1
156 float urightpart [Nx+1];
157 float vrightpart [Nx+1];
158 float u[Nx+1];
159 float v[Nx+1];
160
161 for( int i = 0; i <= Nx; i++){
162     v[ i ] = 0;
163     u[ i ] = 0;
164 }
165
166 int number_of_files = Nt+1;
167
168 char buf[BUFSIZ];
FILE* fp;
169
170 while( count*tau<T){
171
172     for( int i = 0; i<=Nx; i++){
173         for( int k = 0; k<=Ny; k++){
174             fu [ 1 ][ i ][ k ] = runge_kutt(tau, fu [ 0 ][ i ][ k ], fv [ 0 ][ i ][ k ], 1);
175             fv [ 1 ][ i ][ k ] = runge_kutt(tau, fu [ 0 ][ i ][ k ], fv [ 0 ][ i ][ k ], 2);
176         }
177     }
178     for( int k = 0; k<=Ny; k++){
179         urightpart [0] = fu [ 1 ][ 0 ][ k ] + 2*rux*(fu [ 0 ][ 1 ][ k ] - fu [ 0 ][ 0 ][ k ]);
180

```

```

181 vrightpart[0] = fv[1][0][k] + 2*rvx*(fv[0][1][k] - fv[0][0][k]);
182 urightpart[Nx] = fu[1][Nx][k] + 2*rux*(fu[0][Nx-1][k] - fu[0][Nx][k]);
183 vrightpart[Nx] = fv[1][Nx][k] + 2*rvx*(fv[0][Nx-1][k] - fv[0][Nx][k]);
184 for(int i = 1; i < Nx; i++){
185     urightpart[i] = fu[1][i][k]+rux*(fu[0][i+1][k] - 2*fu[0][i][k] + fu[0][i-1][k]);
186     vrightpart[i] = fv[1][i][k]+rvx*(fv[0][i+1][k] - 2*fv[0][i][k] + fv[0][i-1][k]);
187 }
188 solve_tridiagonal(u, Nx+1, aux, bux, cux, urightpart);
189 solve_tridiagonal(v, Nx+1, avx, bvx, cvx, vrightpart);
190 for(int i = 0; i <= Nx; i++){
191     fu[2][i][k] = u[i];
192     fv[2][i][k] = v[i];
193 }
194 }

195 for(int i=0; i<=Nx; i++){
196     urightpart[0] = fu[2][i][0] + 2*ruy*(fu[1][i][1] - fu[1][i][0]);
197     vrightpart[0] = fv[2][i][0] + 2*rvy*(fv[1][i][1] - fv[1][i][0]);
198     urightpart[Ny] = fu[2][i][Ny] + 2*ruy*(fu[1][i][Ny-1] - fu[1][i][Ny]);
199     vrightpart[Ny] = fv[2][i][Ny] + 2*rvy*(fv[1][i][Ny-1] - fv[1][i][Ny]);
200     for(int k = 1; k < Ny; k++){
201         urightpart[k] = fu[2][i][k]+ruy*(fu[1][i][k+1] - 2*fu[1][i][k] + fu[1][i][k-1]);
202         vrightpart[k] = fv[2][i][k]+rvy*(fv[1][i][k+1] - 2*fv[1][i][k] + fv[1][i][k-1]);
203     }
204     solve_tridiagonal(u, Ny+1, auy, buy, cuy, urightpart);
205     solve_tridiagonal(v, Ny+1, avy, bvy, cvy, vrightpart);
206     for(int k = 0; k<=Ny; k++){
207         fu[3][i][k] = u[k];
208         fv[3][i][k] = v[k];
209     }
210 }

211 if(count == 9000){
212     sprintf(buf, "%s%d%s", PREF, count, SUFF);
213     fp = fopen(buf, "w");
214     for(int i = 0; i <= Nx; i++){
215         for(int k = 0; k <= Ny; k++){
216             fprintf(fp, "%f %f %f %f %f \n", count*tau, i*h, k*h, fu[3][i][k], fv[3][i][k]);
217         }
218         fprintf(fp, "\n");
219     }
220     fclose(fp);
221 }
222 }

223 for(int i = 0; i <= Nx; i++){
224     for(int k = 0; k<=Ny; k++){
225         fu[0][i][k] = fu[3][i][k];
226         fv[0][i][k] = fv[3][i][k];
227     }
228 }

229 for(int i= 0; i <BUFSIZ; i++){
230     buf[i] = 0;
231 }
232 count++;
233 }

234 return 0;
235 }

```

Приведем результат исполняемого кода:

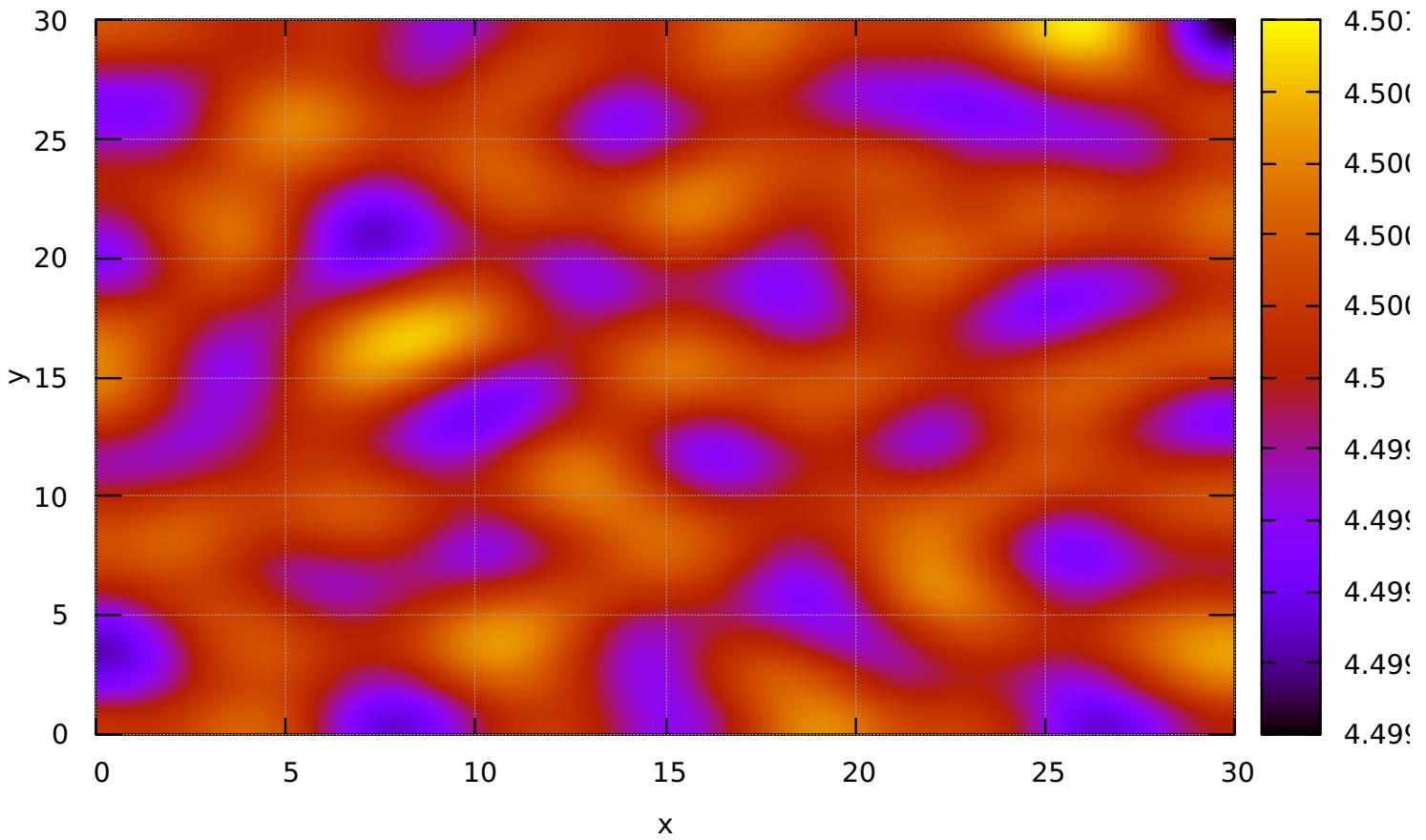


Рис. 21: $u(x,y)$ $A = 4.5$, $B = 7.5$, $D_u = 2$, $D_v = 16$

3.10 Анализ распределенного случая на Wolfram

```

noisefunc[const_, nperturb_: 10, noisefract_: 0.1] :=Module[{ipoints, ridx}, ipoints =
  Table[{xi, const}, {xi, 0.0, 1.0, 0.01}];
  ridx = Table[RandomInteger[{10, 90}], {nperturb}];
  Do[ipoints[[ridx[[ri]]]][[2]] += noisefract*RandomReal[{1, 1}], {ri, nperturb}];
  Return[Interpolation[ipoints, Method -> "Spline"]]];
]

nfu = noisefunc[1.0];
nfv = noisefunc[1.0];

soluv = NDSolve[{D[u[x, t], t] == Du *Derivative[2, 0][u][x, t] + a (b + 1) u[x, t] +
  v[x, t] (u[x, t])^2,
  D[v[x, t], t] == Dv *Derivative[2, 0][v][x, t] + b u[x, t] - v[x, t] (u[x, t])^2,
  u[x, 0] == nfu[x],
  v[x, 0] == nfv[x],
  Derivative[1, 0][u][0, t] == 0,
  Derivative[1, 0][u][L, t] == 0,
  Derivative[1, 0][v][0, t] == 0,
  Derivative[1, 0][v][L, t] == 0} /. {L > 50, Du > 1, Dv > 100, a > 1, b > 1}, {
  u, v}, {x, 0, 50}, {t, 0, 0.05 4000}, MaxSteps -> {15, Infinity}]

```

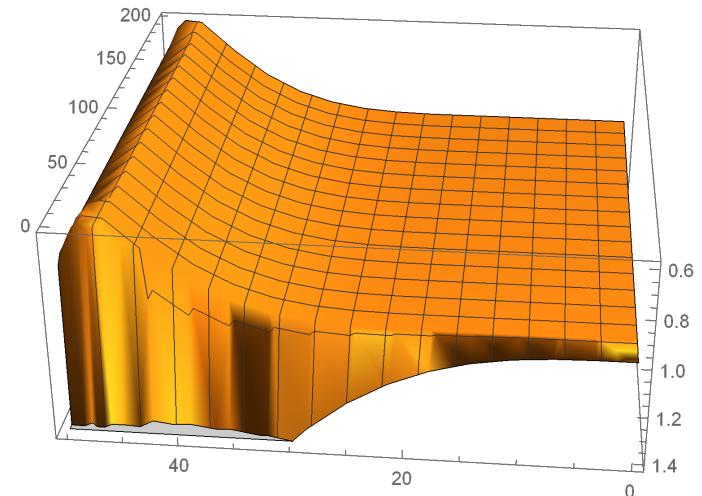
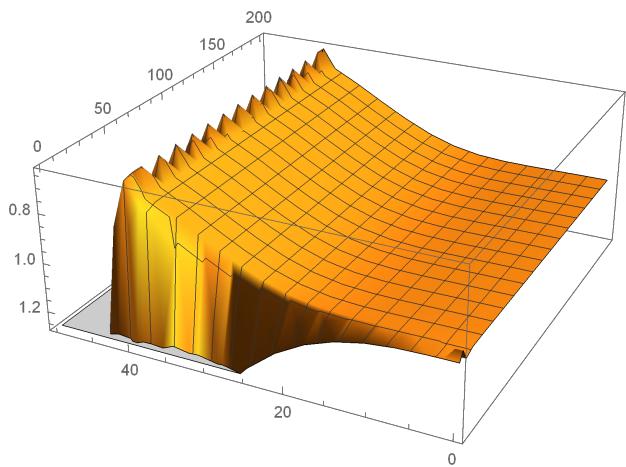


Рис. 22: $u(x,t)$ $A = B=1$, $Du = 1$, $Dv = 100$

Рис. 23: $v(x,t)$ $A = B=1$, $Du = 1$, $Dv = 100$

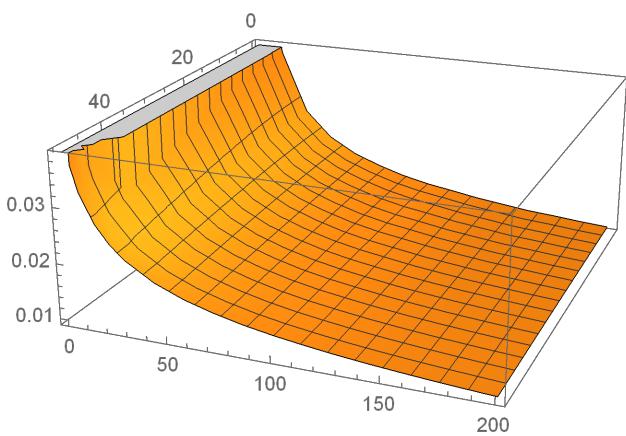


Рис. 24: $u(t)$ $A =1$, $B=1.5$, $Du = 1$, $Dv = 100$

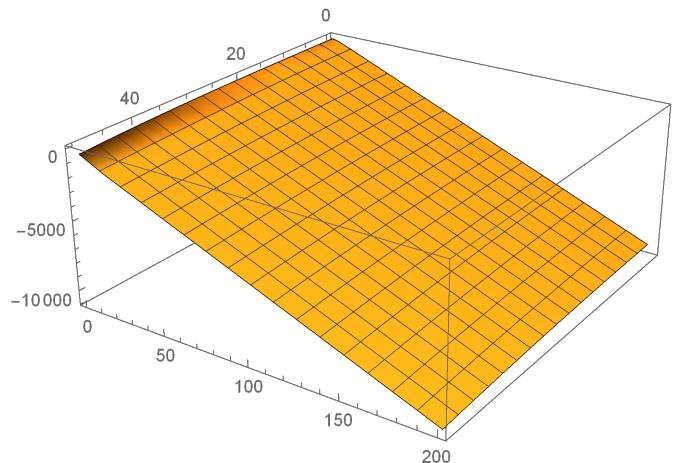


Рис. 25: $u(t)$ $A =1$, $B=1.5$, $Du = 1$, $Dv = 100$