**Министерство науки и вышего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**« Российский химико-технологический университет имени Д.И. Менделеева »**

# ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

## Вариант 7

**Выполнил студент группы КС-46:** **Золотухин Андрей Александрович**

Ссылка на репозиторий: https://github.com/CorgiPuppy/info-proccesses-systems-theory-labs

Принял: Зинченко Дарья Ивановна

Дата сдачи: 26.11.25

**Москва**

**2025**

# Оглавление

# Описание задачи

1. Аналитическим способом найти стационарную точку и определить характер её устойчивости по 1-ому методу Ляпунова;
2. Написать программу, решающую систему ДУ и сттроящую следующие графики:
   1. фазовый портрет системы. Подберите начальные условия, шаг по времени и масштаб таким образом, чтобы тип точки и ее координаты на графике были очевидны (в случае неустойчивой точки начальные условия рекомендуется задавать как можно ближе к неподвижной точке). Число траекторий не меньше восьми!
   2. зависимости $x_1(t)$ и $x_2(t)$ - динамику системы во времени (таким образом, чтобы поведение системы в окрестности неподвижной точки и ее координаты были очевидны.) На одном графике должно быть несколько линий динамики при разных начальных условиях.

## Работа 1.1

В реакторе идеального смешения непрерывного действия протекают реакции по схеме:

$$X \xrightarrow{k_1} Y,$$

$$X \xrightarrow{k_2} P.$$

Математическая модель реактора имеет вид:

$$\begin{cases} \frac{dx}{dt} = \frac{1}{\tau}(x_0 - x) - k_1 x, \\ \frac{dy}{dt} = -\frac{1}{\tau} y + k_1 x - k_2 y. \end{cases}$$

Значения параметров процесса: $x_0 = 6$, $k_1 = 2$, $k_3 = \frac{1}{3}$, $\tau = 1$.
Построить фазовый портрет системы - график в координатах $(x, y)$.
Для решения использовать явную схему Эйлера:

$$\begin{cases} \frac{x^{n+1} - x^n}{\Delta t} = \frac{1}{\tau}(x_0 - x^n) - k_1 x^n, \\ \frac{y^{n+1} - y^n}{\Delta t} = \frac{1}{\tau} y^n + k_1 x^n - k_2 y^n. \end{cases}$$

## Работа 1.2

Система уравнений:

$$\begin{cases} \frac{dx_1}{dt} = x_1 + 5, \\ \frac{dx_2}{dt} = \frac{2}{3} x_2 - 5. \end{cases}$$

Для решения использовать неявную схему Эйлера:

$$\begin{cases} \frac{x^{n+1} - x^n}{\Delta t} = f\left(x_1^{n+1}\right), \\ \frac{y^{n+1} - y^n}{\Delta t} = f\left(x_2^{n+1}\right). \end{cases}$$

## Работа 1.3

Система уравнений:

$$\begin{cases} \frac{dx_1}{dt} = -\frac{2}{3} x_1 - 4, \\ \frac{dx_2}{dt} = \frac{2}{5} x_2. \end{cases}$$

Для решения использовать схему Эйлера:

$$\begin{cases} \frac{x_1^{n+1} - x_1^n}{\Delta t} = f\left(x_1^{n+1}\right), \\ \frac{x_2^{n+1} - x_2^n}{\Delta t} = f(x_2^n). \end{cases}$$

## Работа 1.4

Математическая модель процесса кристаллизации в реакторе (с учётом растворения мелких частиц и кристаллизации крупных) имеет вид:

$$\begin{cases} \frac{d\mu_0}{dt} = k\mu_1 - b + q, \\ \frac{d\mu_1}{dt} = \mu_0(\eta_1 - \eta_2) + d, \end{cases}$$

где $\mu_0$ - нулевой момент функции распределения кристаллов по размерам, характеризующий общее количество частиц в единице объёма реактора; $\mu_1$ - первый момент функции распределения, характеризующий суммарный линейный размер кристаллов; $k$ - константа скорости образования зародышей; $k\mu_1$ - скорость образования зародышей; $b$ - скорость отбора зародышей; $q$ - скорость пополнения крупными частицами; $\eta_1$ - скорость роста кристаллов; $\eta_2$ - скорость растворения кристаллов; $d$ - суммарный линейный размер поступающих частиц.
Значения параметров процесса: $k = \frac{3}{7}$, $b = \frac{9}{2}$, $q = 3$, $\eta_1 = \frac{3}{7}$, $\eta = 1$, $\mathrm{d} = 2$.
Для решения использовать явную схему Эйлера:

$$\begin{cases} \frac{\mu_0^{n+1} - \mu_0^n}{\Delta t} = k\mu_1^n - b + q, \\ \frac{\mu_1^{n+1} - \mu_1^n}{\Delta t} = \mu_0^n(\eta_1 - \eta_2) + d. \end{cases}$$

## Работа 1.5

Система уравнений:

$$\begin{cases} \frac{dx_1}{dt} = -2x_1 - 2x_2 + 6, \\ \frac{dx_2}{dt} = 3x_1 - 3x_2. \end{cases}$$

Для решения использовать схему Эйлера:

$$\begin{cases} \frac{x_1^{n+1} - x_1^n}{\Delta t} = f\big(x_1^{n+1}, x_2^n\big), \\ \frac{x_2^{n+1} - x_2^n}{\Delta t} = f\big(x_1^n, x_2^{n+1}\big). \end{cases}$$

## Работа 1.6

Система уравнений:

$$\begin{cases} \frac{dx_1}{dt} = 333x_1 - x_2 + 2, \\ \frac{dx_2}{dt} = 4x_1 + 3x_2 + 7. \end{cases}$$

Для решения использовать схему Эйлера:

$$\begin{cases} \frac{x_1^{n+1} - x_1^n}{\Delta t} = f\big(x_1^{n+1}, x_2^n\big), \\ \frac{x_2^{n+1} - x_2^n}{\Delta t} = f\big(x_1^n, x_2^{n+1}\big). \end{cases}$$

# Выполнение задачи

## Работа 1.1

### Аналитическое решение

На Рис. 1 представлено аналитическое решение.

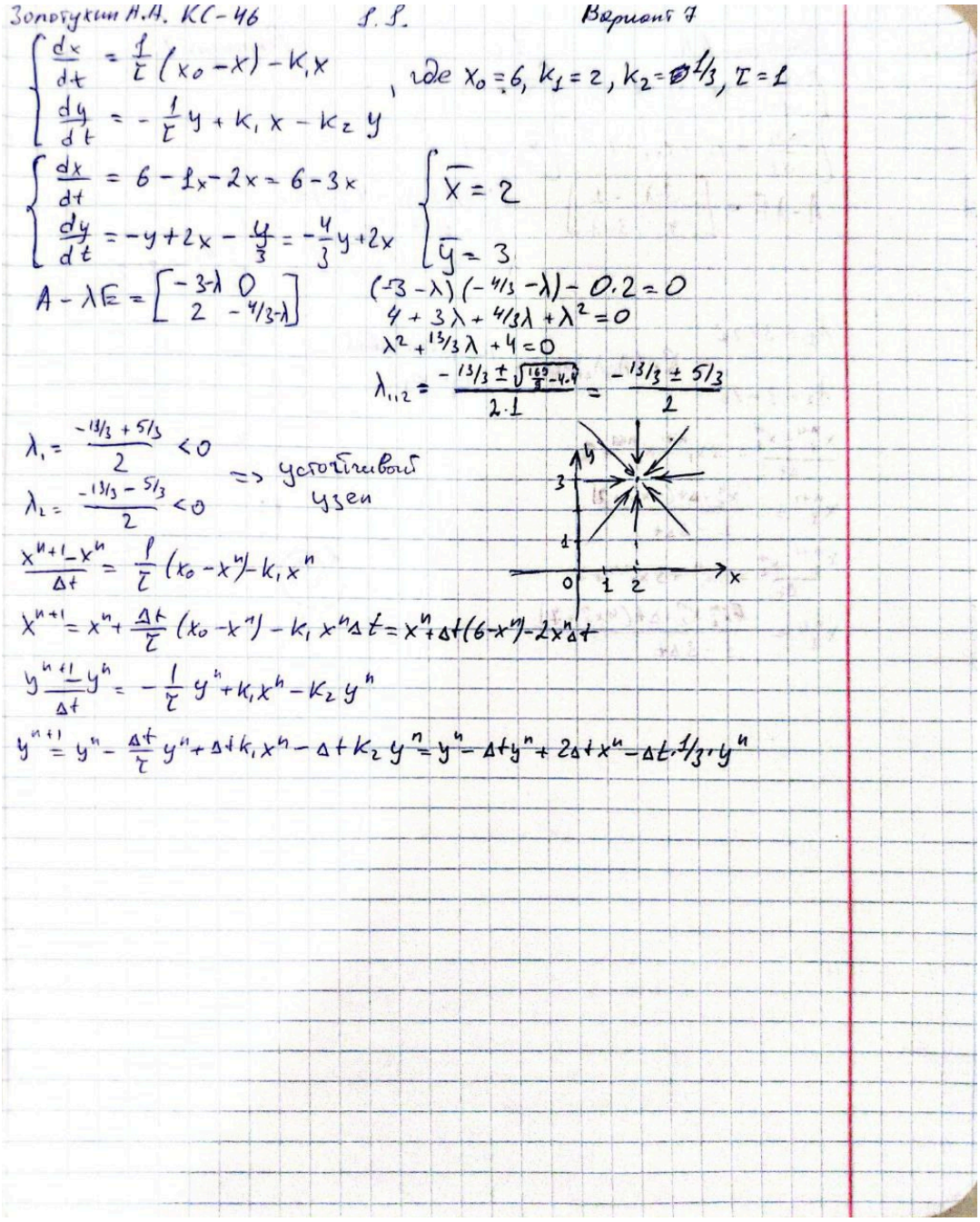


Рис. 1: Аналитическое решение.

### Алгоритм программы

### Графики

На Рис. 2 представлен фазовый портрет.

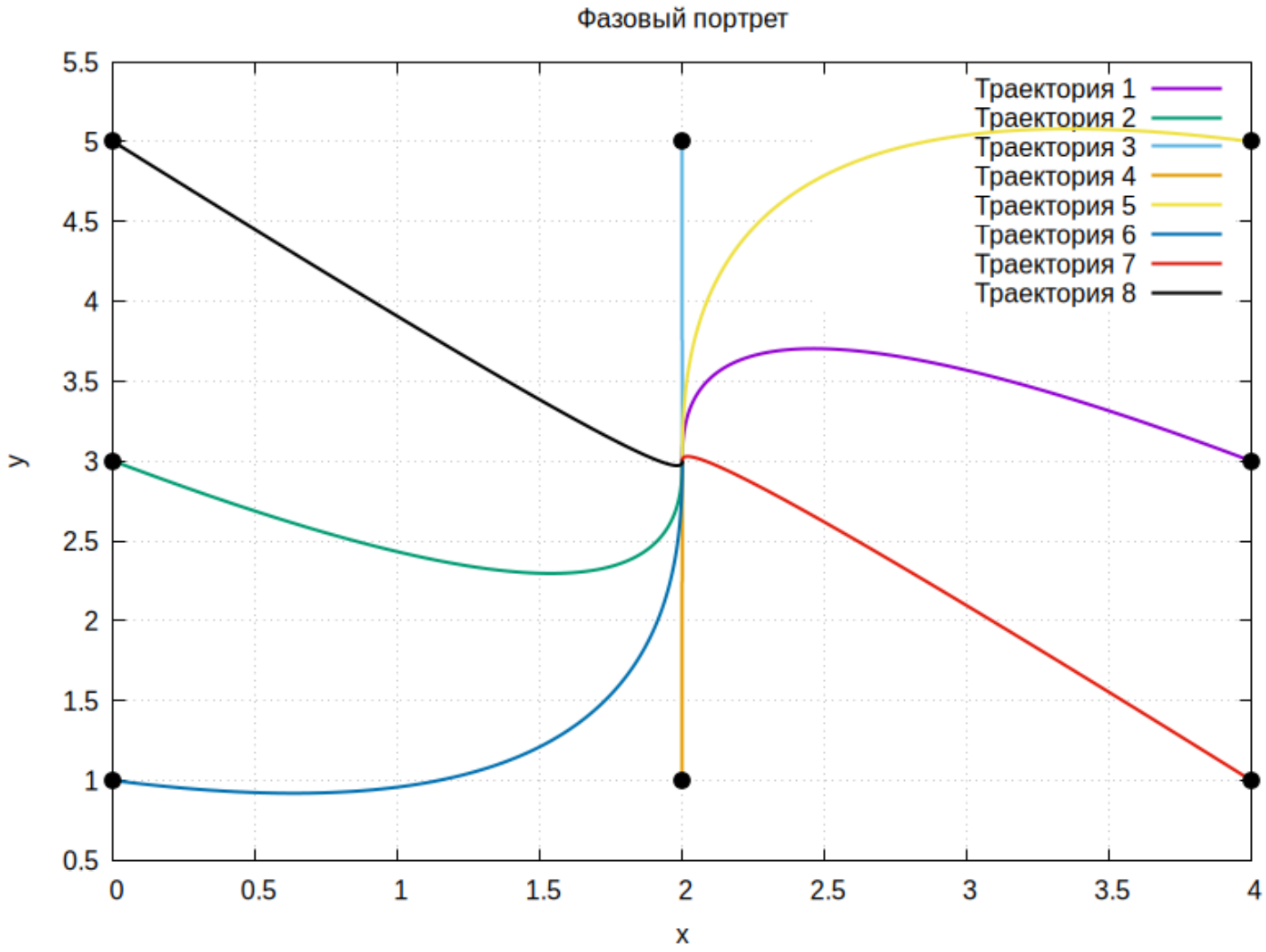


Рис. 2: Фазовый портрет.

На Рис. 3 представлены зависимости x(t) и y(t).

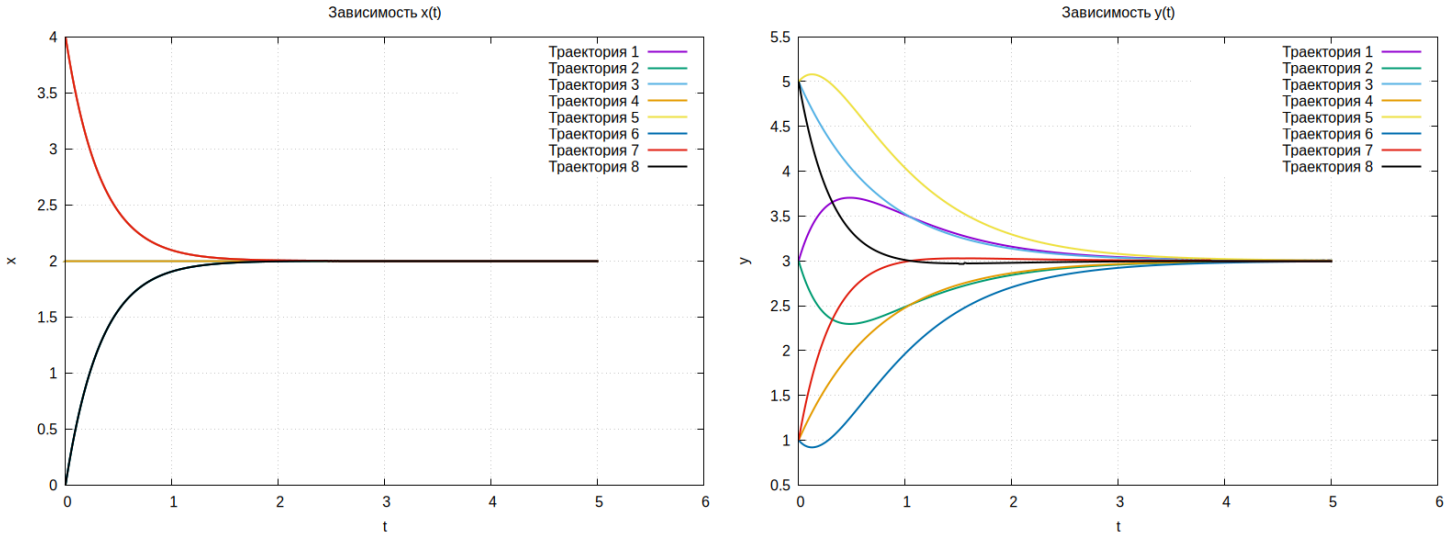


Рис. 3: Зависимости x(t) и y(t).

# Работа 1.2

## Аналитическое решение

На Рис. 4 представлено аналитическое решение.

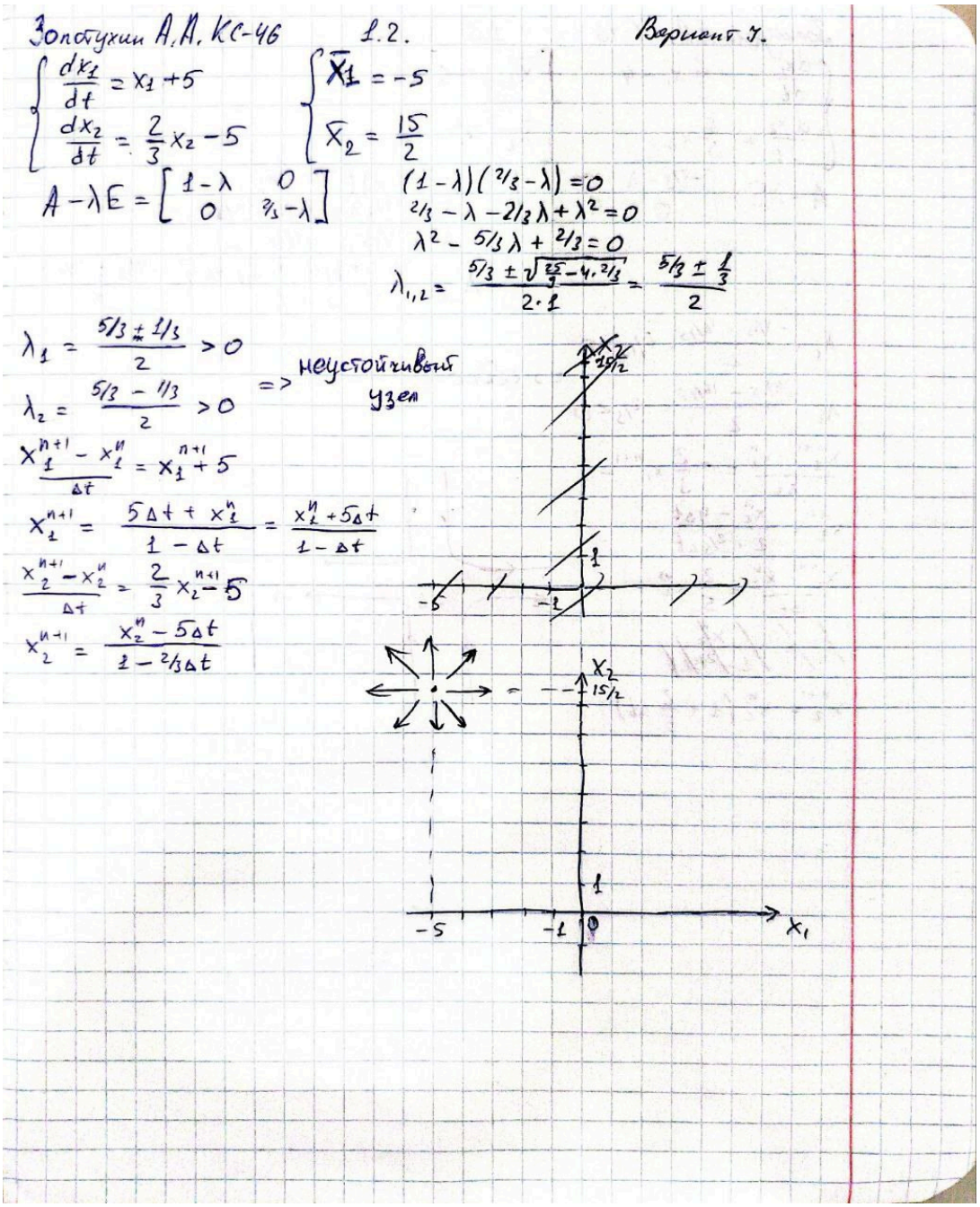


Рис. 4: Аналитическое решение.

## Алгоритм программы

## Графики

На Рис. 5 представлен фазовый портрет.

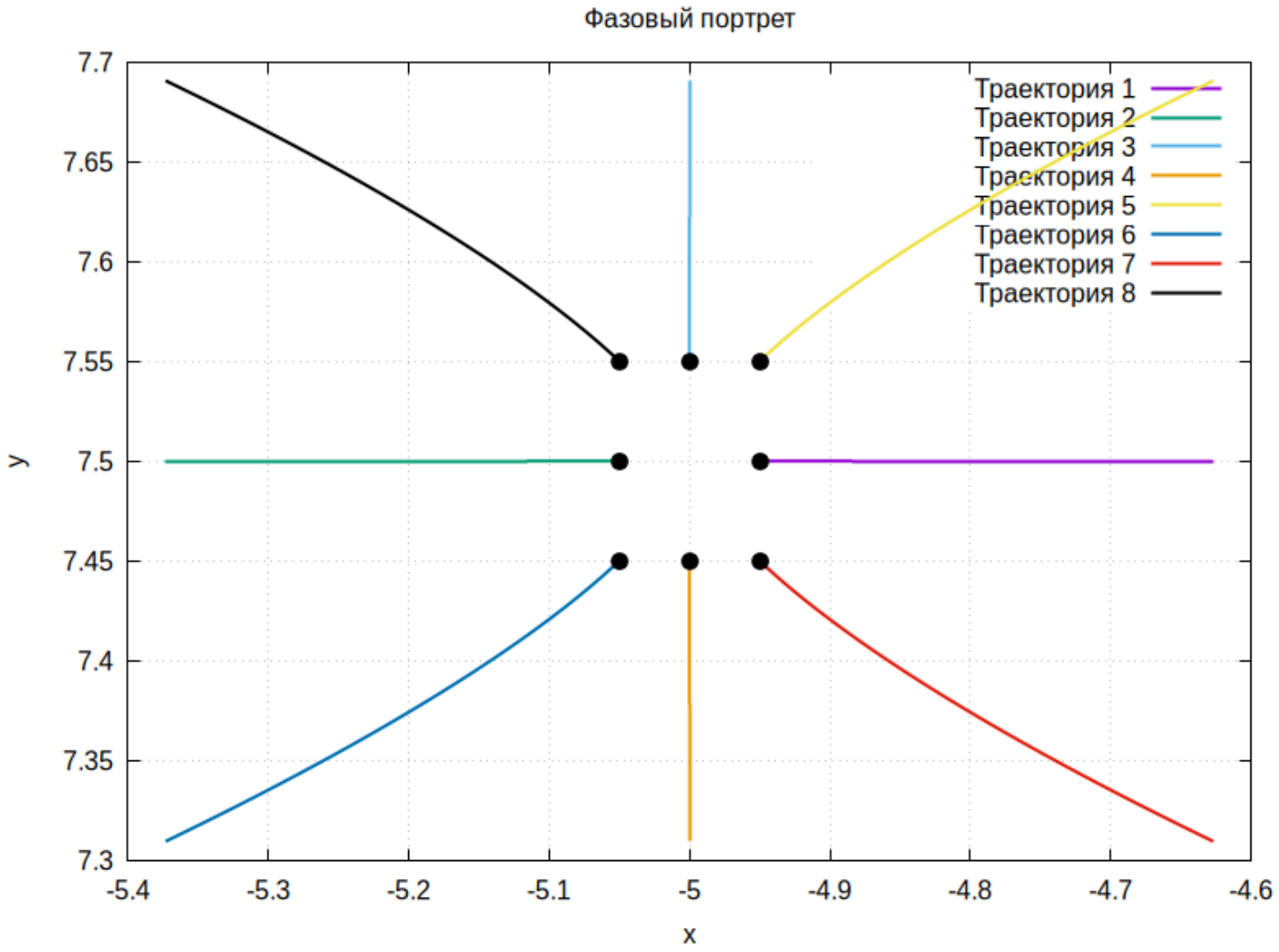


Рис. 5: Фазовый портрет.

На Рис. 6 представлены зависимости x(t) и y(t).

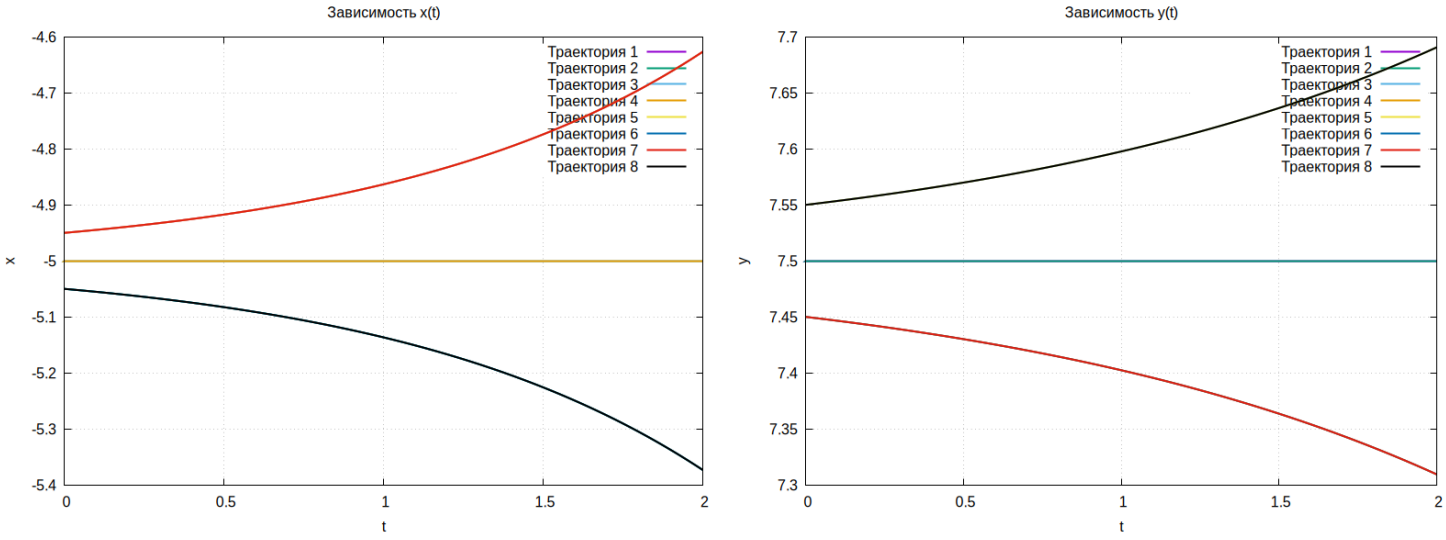


Рис. 6: Зависимости x(t) и y(t).

# Работа 1.3

## Аналитическое решение

На Рис. 7 представлено аналитическое решение.

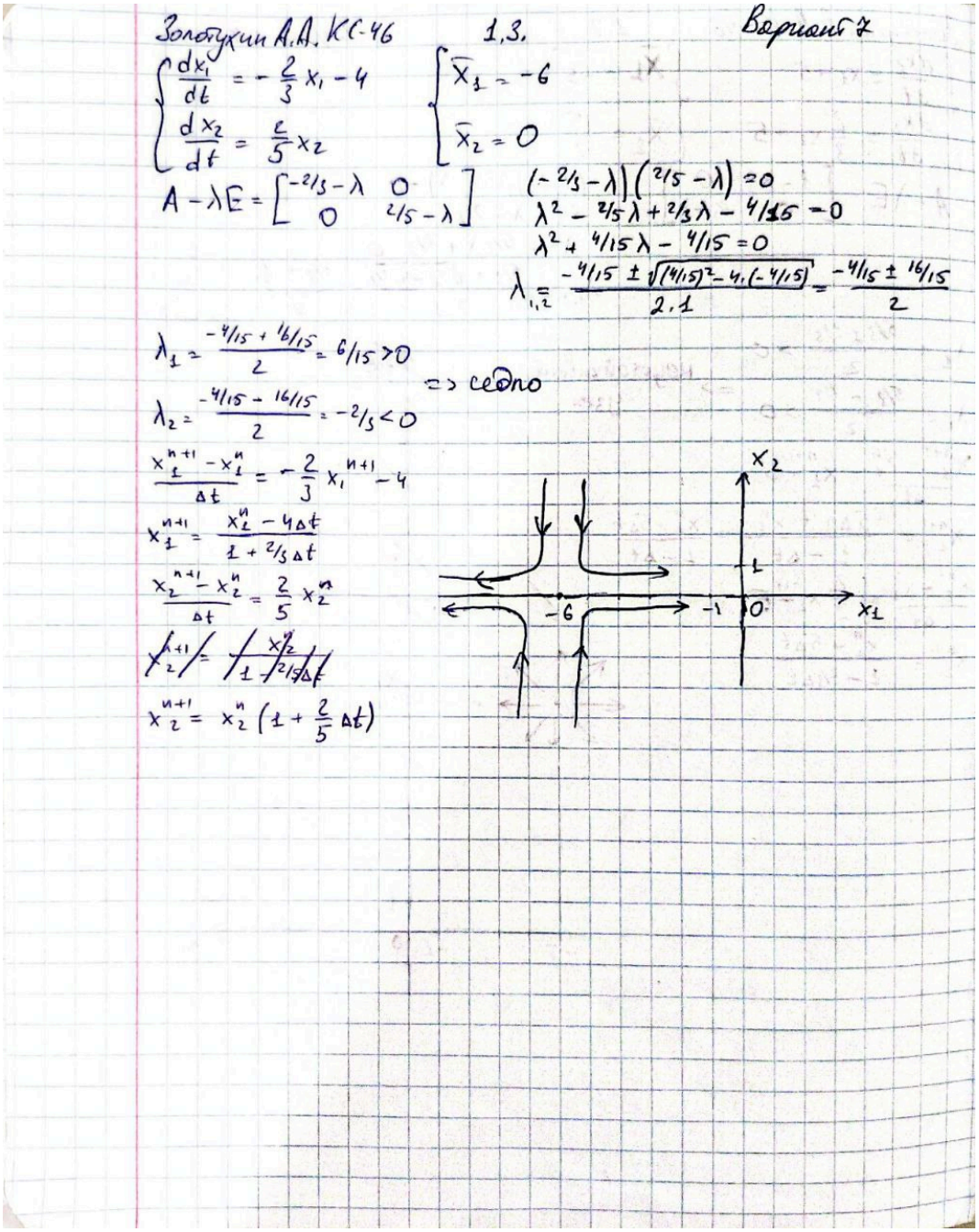


Рис. 7: Аналитическое решение.

## Алгоритм программы

## Графики

На Рис. 8 представлен фазовый портрет.

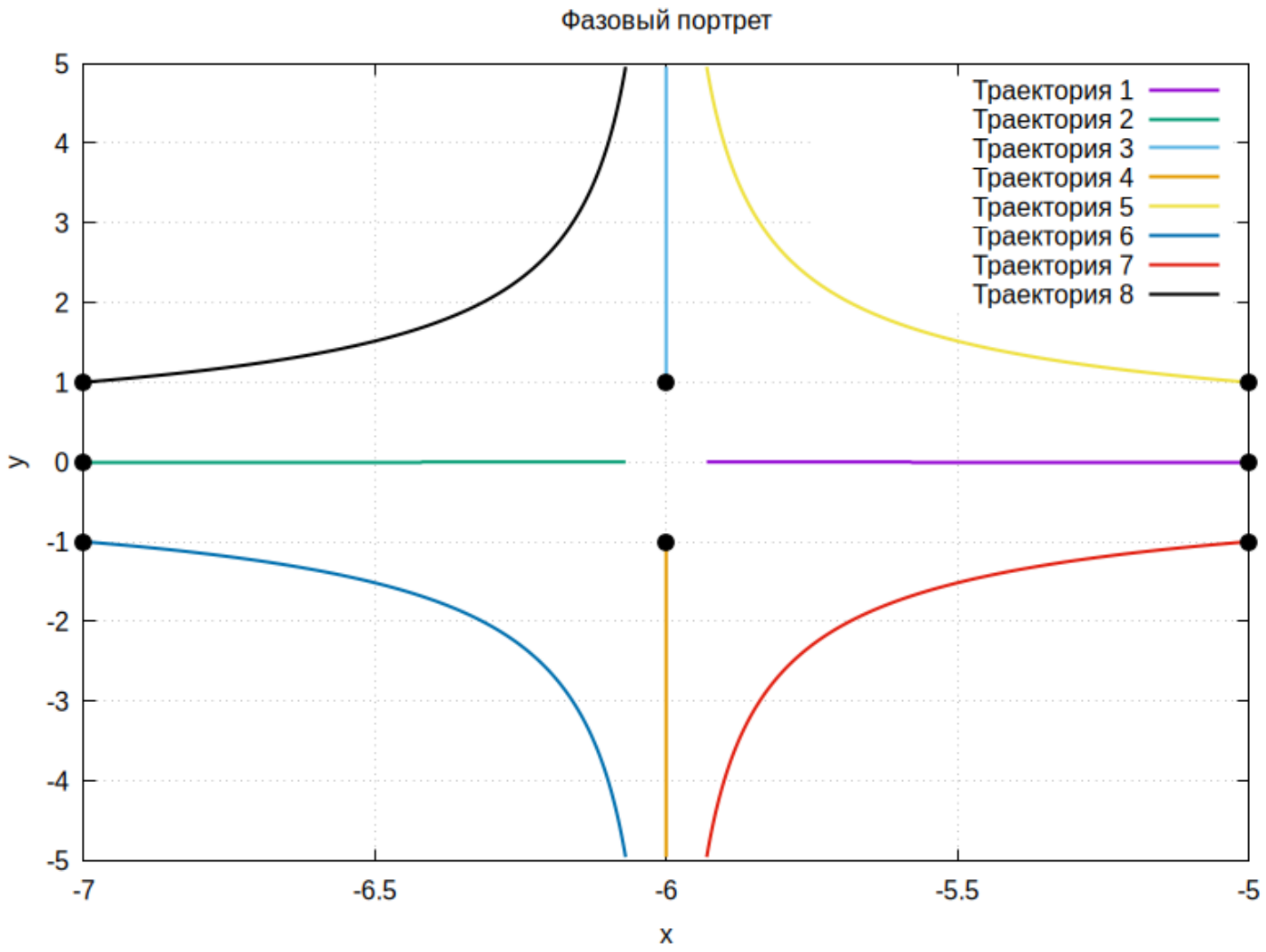


Рис. 8: Фазовый портрет.

На Рис. 9 представлены зависимости x(t) и y(t).

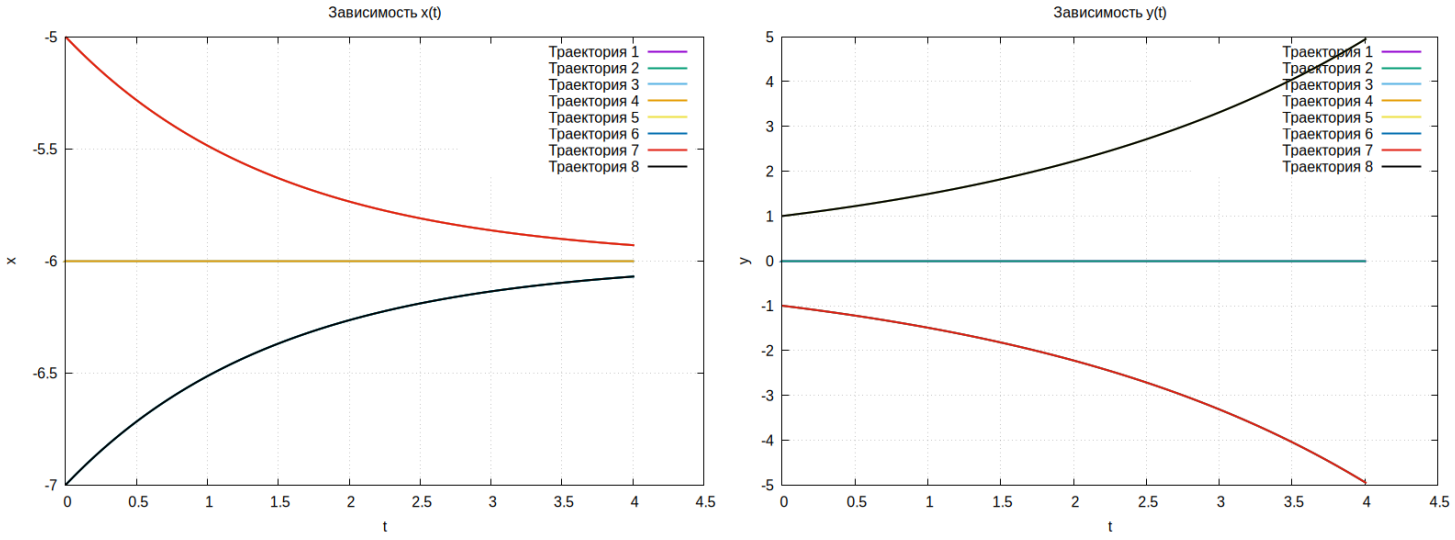


Рис. 9: Зависимости x(t) и y(t).

# Работа 1.4

## Аналитическое решение

На Рис. 10 представлено аналитическое решение.

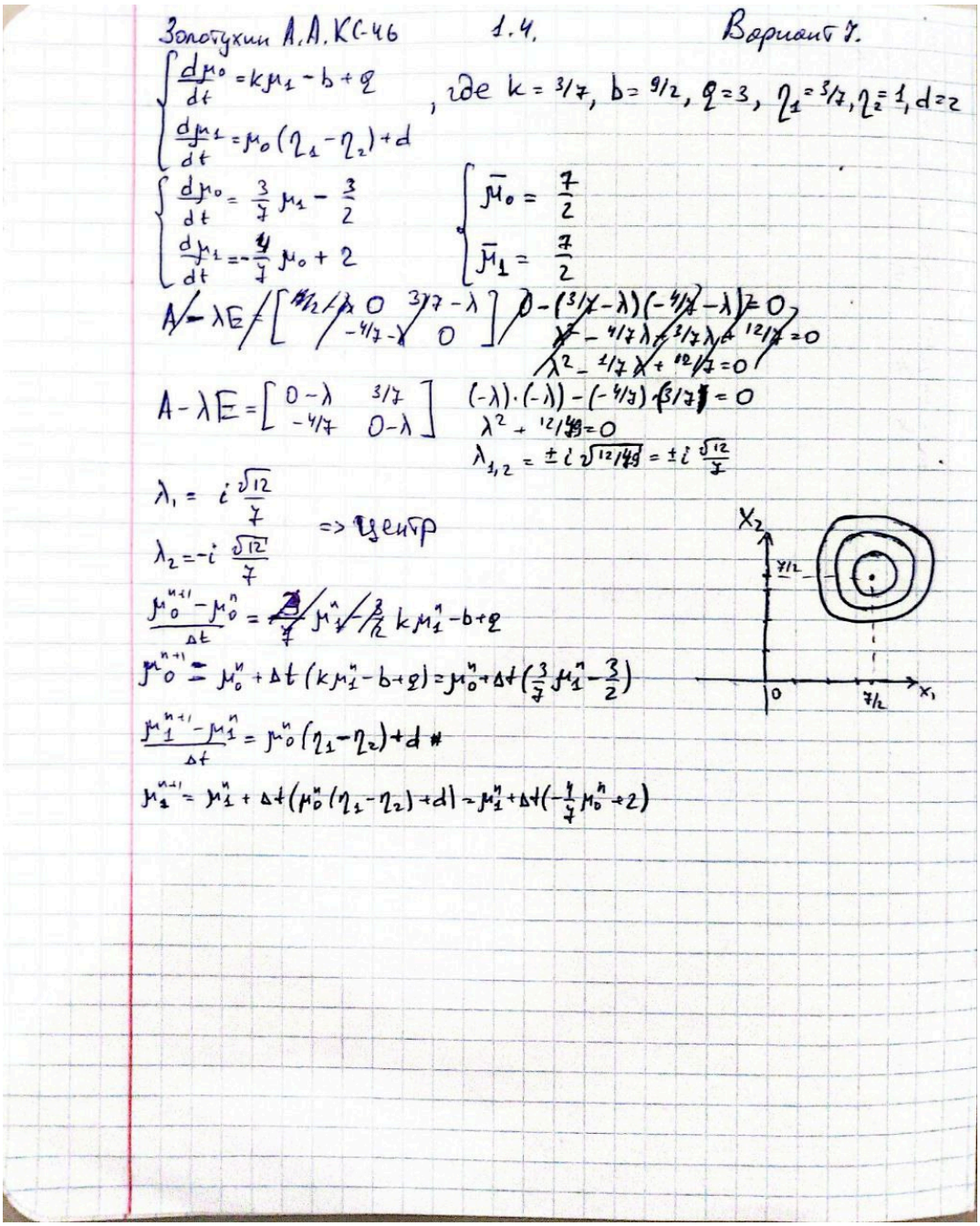


Рис. 10: Аналитическое решение.

## Алгоритм программы

## Графики

На Рис. 11 представлен фазовый портрет.

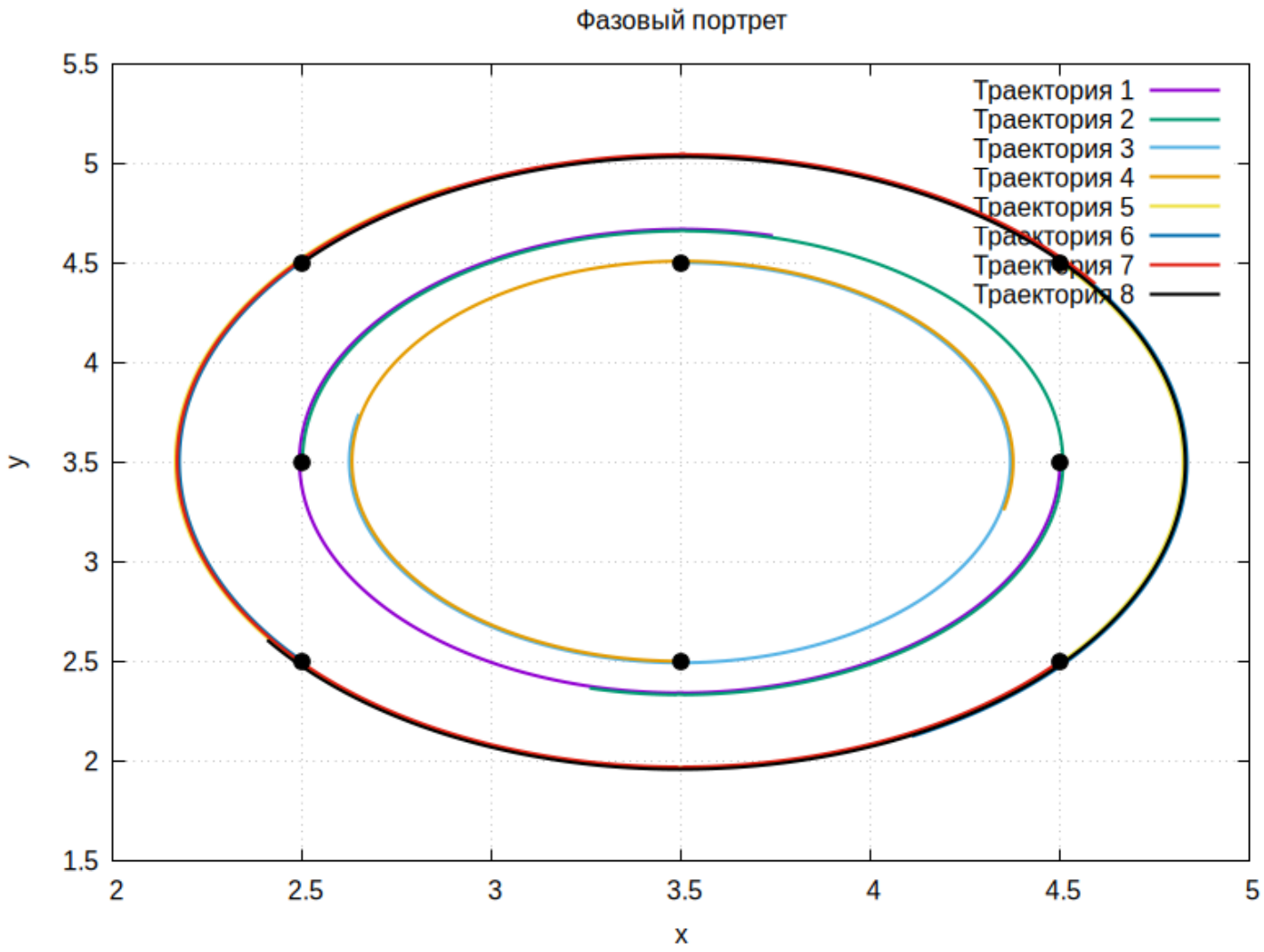


Рис. 11: Фазовый портрет.

На Рис. 12 представлены зависимости x(t) и y(t).

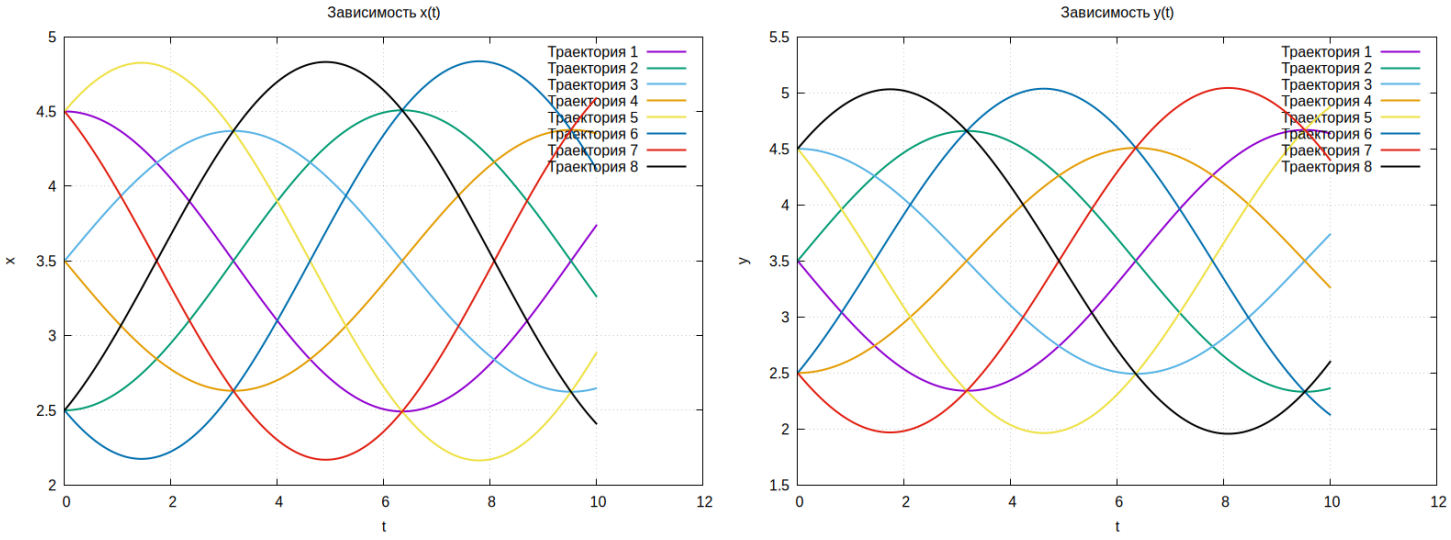


Рис. 12: Зависимости x(t) и y(t).

# Работа 1.5

## Аналитическое решение

На Рис. 13 представлено аналитическое решение.

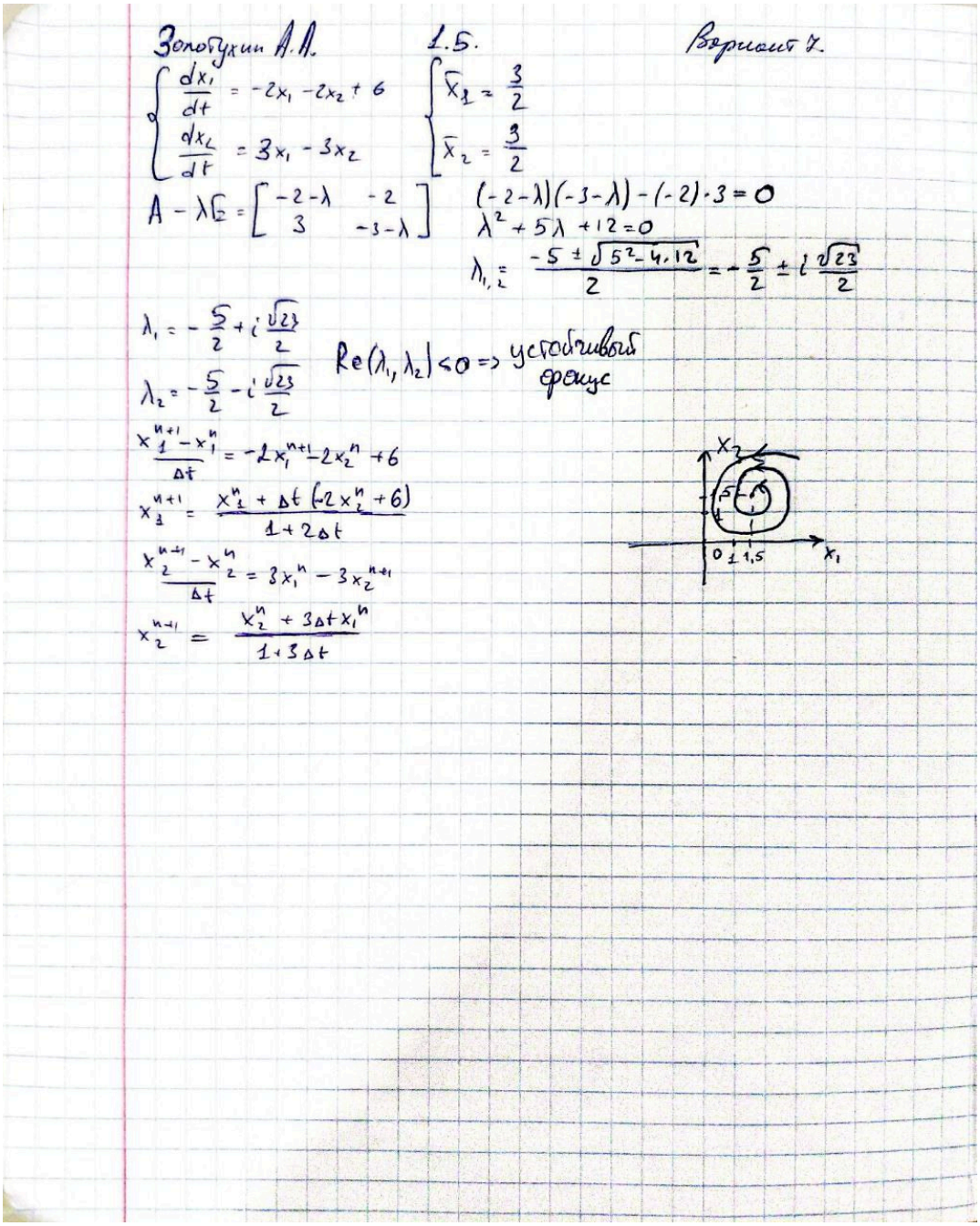


Рис. 13: Аналитическое решение.

## Алгоритм программы

## Графики

На Рис. 14 представлен фазовый портрет.

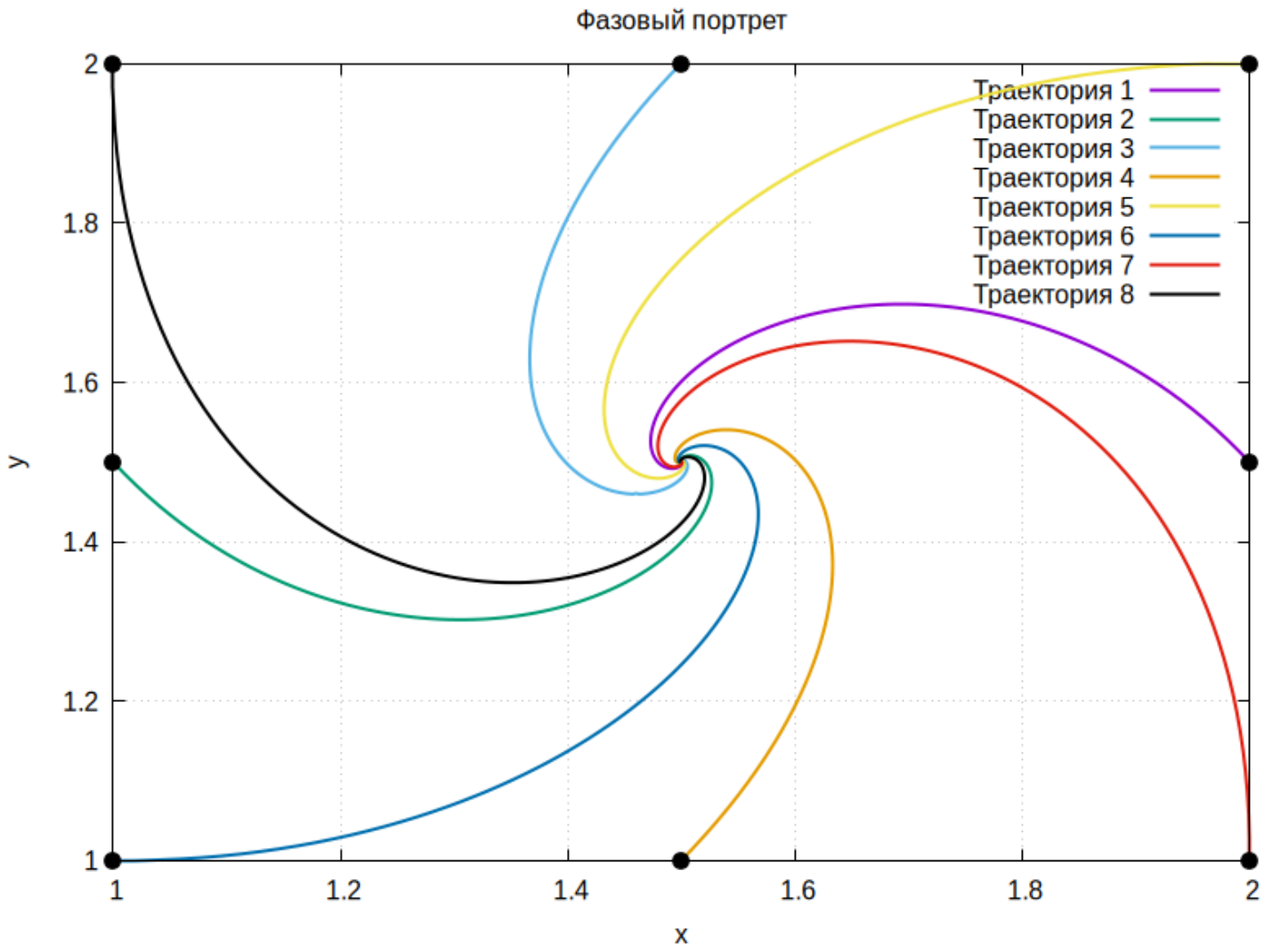


Рис. 14: Фазовый портрет.

На Рис. 15 представлены зависимости x(t) и y(t).

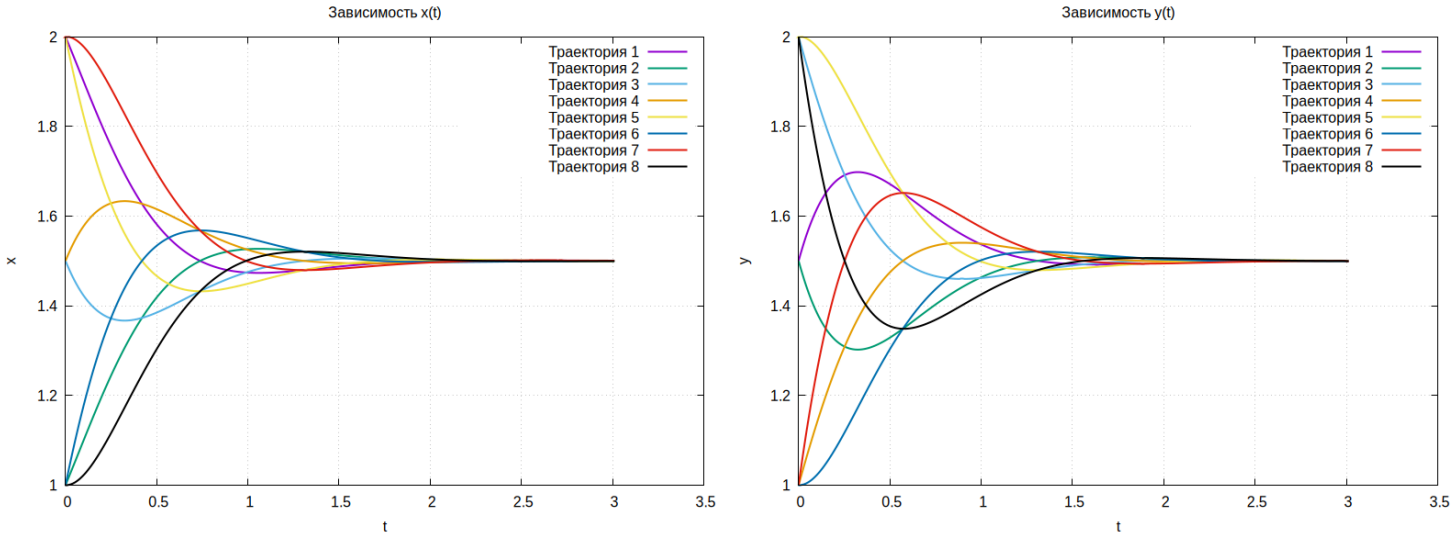


Рис. 15: Зависимости x(t) и y(t).

# Работа 1.6

## Аналитическое решение

На Рис. 16 представлено аналитическое решение.

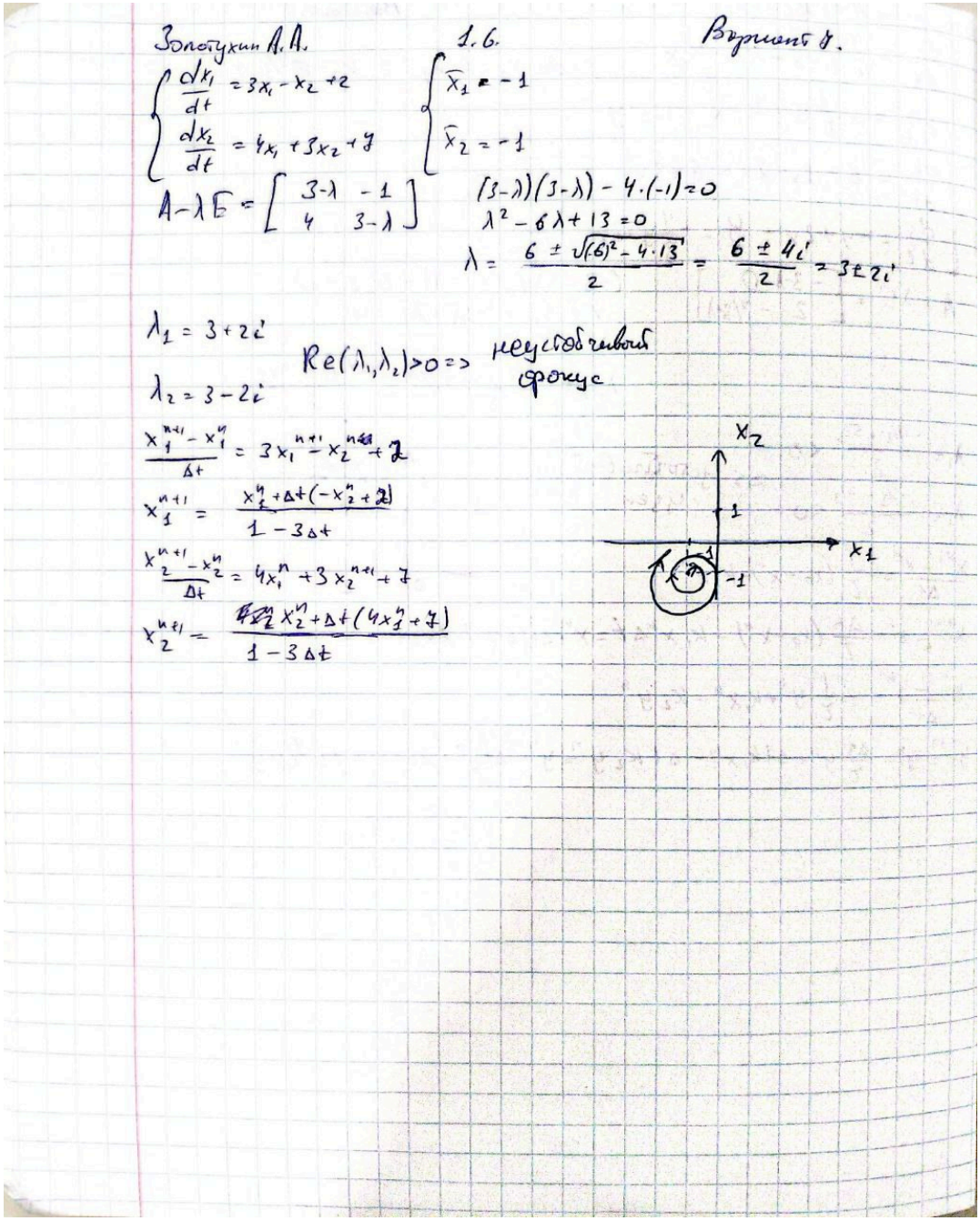


Рис. 16: Аналитическое решение.

## Алгоритм программы

## Графики

На Рис. 17 представлен фазовый портрет.

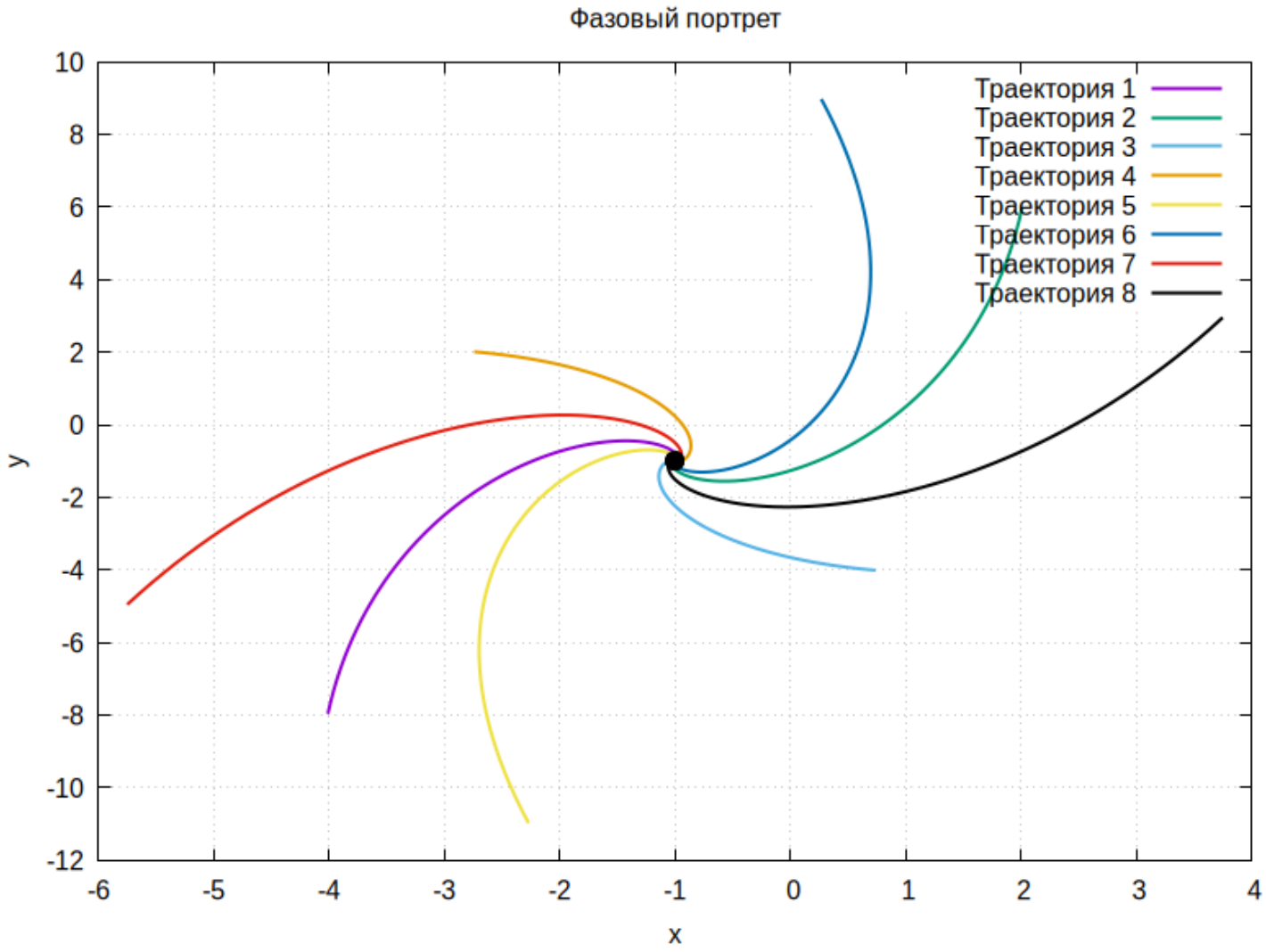


Рис. 17: Фазовый портрет.

На Рис. 18 представлены зависимости x(t) и y(t).

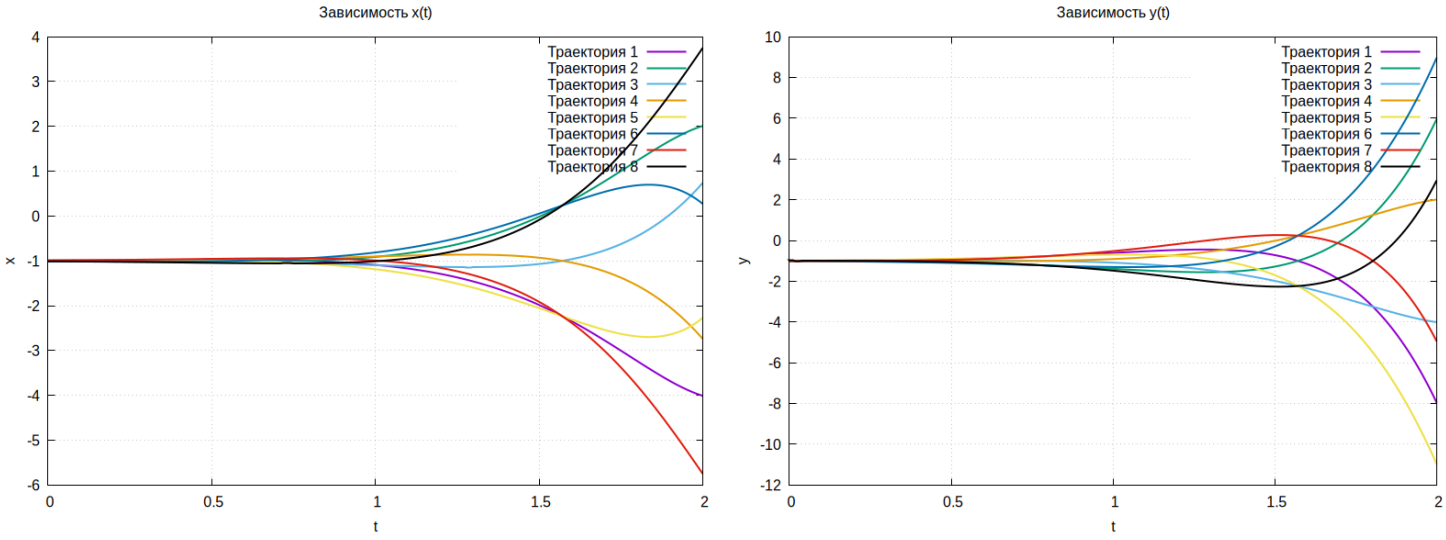


Рис. 18: Зависимости x(t) и y(t).

# Код

```
#ifndef CONSTANTS_H
#define CONSTANTS_H

#include <string>

namespace Constants {
  const std::string DAT_DIR = "plots/dat-files/";

  namespace Task1 {
    const double x0 = 6.0;
    const double k1 = 2.0;
    const double k2 = 1.0 / 3.0;
    const double tau = 1.0;

    const double dt = 0.01;
    const double T = 5.0;
    const std::string filename = "task1.dat";
  }

  namespace Task2 {
    const double dt = 0.01;
    const double T = 2.0;
    const std::string filename = "task2.dat";
  }

  namespace Task3 {
    const double dt = 0.01;
    const double T = 4.0;
    const std::string filename = "task3.dat";
  }

  namespace Task4 {
    const double k = 3.0 / 7.0;
    const double b = 9.0 / 2.0;
    const double q = 3.0;
    const double eta1 = 3.0 / 7.0;
    const double eta2 = 1.0;
    const double d = 2.0;

    const double dt = 0.01;
    const double T = 10.0;
    const std::string filename = "task4.dat";
  }

  namespace Task5 {
    const double dt = 0.01;
    const double T = 3.0;
    const std::string filename = "task5.dat";
  }

  namespace Task6 {
    const double dt = 0.01;
    const double T = 2;
    const std::string filename = "task6.dat";
  }
}

#endif
```

```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <fstream>
#include <iomanip>

#include "Constants.h"

struct Point {
  double x;
  double y;
};

void saveToDat(const std::string&, const std::vector<std::vector<Point>>&, double);
std::vector<Point> generateInitialConditions(double, double, double);
void solveTask1();
void solveTask2();
void solveTask3();
void solveTask4();
void solveTask5();
void solveTask6();

int main() {
  solveTask1();
  solveTask2();
  solveTask3();
  solveTask4();
  solveTask5();
  solveTask6();

  return 0;
}

void saveToDat(const std::string& filename, const std::vector<std::vector<Point>>& trajectories,
double dt) {
  std::string path = Constants::DAT_DIR + filename;
  std::ofstream file(path);

  if (!file.is_open()) {
    std::cerr << "Ошибка: не удалось открыть файл " << path << std::endl;
    return;
  }

  file << std::fixed << std::setprecision(6);

  for (std::vector traj : trajectories) {
    for (long unsigned i = 0; i < traj.size(); i++) {
      double t = i * dt;
      file << t << " " << traj[i].x << " " << traj[i].y << "\n";
    }
    file << "\n\n";
  }

  file.close();
}

std::vector<Point> generateInitialConditions(double xs, double ys, double epsilon) {
```

```
  return {
    {xs + epsilon, ys}, {xs - epsilon, ys},
    {xs, ys + epsilon}, {xs, ys - epsilon},
    {xs + epsilon, ys + epsilon}, {xs - epsilon, ys - epsilon},
    {xs + epsilon, ys - epsilon}, {xs - epsilon, ys + epsilon}
  };
}

void solveTask1() {
  using namespace Constants::Task1;

  double xs = x0 / (1.0 + k1 * tau);
  double ys = (k1 * xs) / (1.0/tau + k2);

  std::vector<Point> starts = generateInitialConditions(xs, ys, 2.0);
  std::vector<std::vector<Point>> trajectories;

  for (Point p : starts) {
    std::vector<Point> traj;
    traj.push_back(p);
    double x = p.x;
    double y = p.y;
    for (double t = 0; t < T; t += dt) {
      double dx = dt * (1.0/tau * (x0 - x) - k1 * x);
      double dy = dt * (-1.0/tau * y + k1 * x - k2 * y);

      x += dx;
      y += dy;

      traj.push_back({x, y});
    }
    trajectories.push_back(traj);
  }

  saveToDat(filename, trajectories, dt);
}

void solveTask2() {
  using namespace Constants::Task2;

  double x1s = -5.0;
  double x2s = 15.0 / 2.0;

  std::vector<Point> starts = generateInitialConditions(x1s, x2s, 0.05);
  std::vector<std::vector<Point>> trajectories;

  for (Point p : starts) {
    std::vector<Point> traj;
    traj.push_back(p);
    double x1 = p.x;
    double x2 = p.y;
    for (double t = 0; t < T; t += dt) {
      x1 = (x1 + 5.0 * dt) / (1.0 - dt);
      x2 = (x2 - 5.0 * dt) / (1.0 - (2.0/3.0) * dt);

      traj.push_back({x1, x2});
    }
    trajectories.push_back(traj);
  }
```

```
  saveToDat(filename, trajectories, dt);
}

void solveTask3() {
  using namespace Constants::Task3;

  double x1s = -6.0;
  double x2s = 0.0;

  std::vector<Point> starts = generateInitialConditions(x1s, x2s, 1.0);
  std::vector<std::vector<Point>> trajectories;

  for (Point p : starts) {
    std::vector<Point> traj;
    traj.push_back(p);
    double x1 = p.x;
    double x2 = p.y;
    for (double t = 0; t < T; t += dt) {
      x1 = (x1 - 4.0 * dt) / (1.0 + (2.0/3.0) * dt);
      x2 = x2 * (1.0 + (2.0/5.0) * dt);

      traj.push_back({x1, x2});
    }
    trajectories.push_back(traj);
  }

  saveToDat(filename, trajectories, dt);
}

void solveTask4() {
  using namespace Constants::Task4;

  double mu1s = (b - q) / k;
  double mu0s = -d / (eta1 - eta2);

  std::vector<Point> starts = generateInitialConditions(mu0s, mu1s, 1.0);
  std::vector<std::vector<Point>> trajectories;

  for (Point p : starts) {
    std::vector<Point> traj;
    traj.push_back(p);
    double mu0 = p.x;
    double mu1 = p.y;
    for (double t = 0; t < T; t += dt) {
      double dmu0 = dt * (k * mu1 - b + q);
      double dmu1 = dt * (mu0 * (eta1 - eta2) + d);

      mu0 += dmu0;
      mu1 += dmu1;

      traj.push_back({mu0, mu1});
    }
    trajectories.push_back(traj);
  }

  saveToDat(filename, trajectories, dt);
}

void solveTask5() {
  using namespace Constants::Task5;
```

```
  double x1s = 3.0/2.0;
  double x2s = 3.0/2.0;

  std::vector<Point> starts = generateInitialConditions(x1s, x2s, 0.5);
  std::vector<std::vector<Point>> trajectories;

  for (Point p : starts) {
    std::vector<Point> traj;
    traj.push_back(p);
    double x1 = p.x;
    double x2 = p.y;
    for (double t = 0; t < T; t += dt) {
      double x1_old = x1;
      double x2_old = x2;

      x1 = (x1_old + dt * (-2.0 * x2_old + 6.0)) / (1.0 + 2.0 * dt);
      x2 = (x2_old + dt * 3.0 * x1_old) / (1.0 + dt * 3.0);

      traj.push_back({x1, x2});
    }
    trajectories.push_back(traj);
  }

  saveToDat(filename, trajectories, dt);
}

void solveTask6() {
  using namespace Constants::Task6;

  double x1s = -1.0;
  double x2s = -1.0;

  std::vector<Point> starts = generateInitialConditions(x1s, x2s, 0.01);
  std::vector<std::vector<Point>> trajectories;

  for (Point p : starts) {
    std::vector<Point> traj;
    traj.push_back(p);
    double x1 = p.x;
    double x2 = p.y;
    for (double t = 0; t < T; t += dt) {
      double x1_old = x1;
      double x2_old = x2;

      x1 = (x1_old + dt * (-x2_old + 2.0)) / (1.0 - 3.0 * dt);
      x2 = (x2_old + dt * (4.0 * x1_old + 7.0)) / (1.0 - dt * 3.0);

      traj.push_back({x1, x2});
    }
    trajectories.push_back(traj);
  }

  saveToDat(filename, trajectories, dt);
}
```

# Выводы