Name: Artur

Surname: Mukhutdinov

Group number: CS-03

e-mail: a.mukhutdinov@innopolis.university

Github link: https://github.com/CatOrLeader/Linear-Algebra

cout << shortRepontOnDiffEqs << endl;

The main implementation of the solution of *Differential Equations using predator-prey* algorithm. By the way, the names of the function and variables describe themselves, so for now I attach only the int main() function from my implementation. (There are can be some changes in code during the plotting procedure to, for instance, make LEGEND visible if graphs are intersect with it)

A set of points will be taken from the internet sources to take the most interesting example of the model. (Idea taken from the Lotka-Volterra)

```
#ifdef WIN64
#define GNUPLOT_NAME "D:\\Studying\\gnuplot\\bin\\gnuplot -persist"
#else
#define GNUPLOT_NAME "gnuplot -persist"
#endif

int main() {

#ifdef WIN64
    FILE *plotter = _popen(GNUPLOT_NAME, "w");
#else
    ...
#endif

    if (plotter == nullptr) {
        return -1;
    }

    int victimsNum;
    int killersNum;
    double alpha_1, alpha_2;
    double betta_1, betta_2;
    double timeLimit;
    int pointsNum;
```

```cpp
// Parsing data
cin >> victimsNum >> killersNum;
cin >> alpha_1 >> betta_1;
cin >> alpha_2 >> betta_2;
cin >> timeLimit;
cin >> pointsNum;

// Output format
cout << setprecision(2) << fixed;
cout << showpoint;

// Create and output array of time moments
vector<double> moments;
double temp = 0.0;
double step = timeLimit / pointsNum;

cout << "t:" << endl;
while (temp <= timeLimit) {
    moments.emplace_back(temp);
    cout << temp << " ";
    temp += step;
}
cout << endl;
```

```
// Create and output array of victims v(t_{i}) entitled v:
vector<double> victims;
temp = 0.0;

cout << "v:" << endl;
for (int i = 0; i < moments.size(); i++) {
    victims.emplace_back(v_ti(victimsNum, killersNum, alpha_1, alpha_2, betta_1, betta_2, moments.at(i)));
    cout << victims.at(i) << " ";
}
cout << endl;

// Create and output array of killers k(t_{i}) entitled v:
vector<double> killers;
temp = 0.0;

cout << "k:" << endl;
for (int i = 0; i < moments.size(); i++) {
    killers.emplace_back(k_ti(victimsNum, killersNum, alpha_1, alpha_2, betta_1, betta_2, moments.at(i)));
    cout << killers.at(i) << " ";
}
```

```
// Formatting plot
fprintf(plotter, "%s\n", "set border linewidth 1.5\n"
                         "set style line 1 linecolor rgb '#0060ad' linetype 1 linewidth 2\n"
                         "set style line 2 linecolor rgb '#dd181f' linetype 1 linewidth 2");

// Formatting legend
fprintf(plotter, "%s\n", "set key at 50,7\n"
                         "set xlabel 't'\n"
                         "set ylabel 'v'\n"
                         "set xrange [0:50]\n"
                         "set yrange [0:7]\n"
                         "set xtics 5\n"
                         "set ytics 0.5\n"
                         "set tics scale 0.75");
```

```
    // CHOOSE YOUR FIGHTER (obviously, graph for the plotting) :)

    // Output v(t) and k(t)

//    fprintf(plotter, "%s\n", "plot '-' title 'v(t)' with lines linestyle 1, '-' title 'k(t)' with lines linestyle 2");
//    for (int i = 0; i < pointsNum; i++) {
//        fprintf(plotter, "%lf %lf\n", moments[i], victims[i]);
//    }
//    fprintf(plotter, "%c\n", 'e');
//    for (int i = 0; i < pointsNum; i++) {
//        fprintf(plotter, "%lf %lf\n", moments[i], killers[i]);
//    }
//    fprintf(plotter, "%c\n", 'e');

    // Output v(k)

//    fprintf(plotter, "%s\n", "plot '-' title 'v(k)' with lines linestyle 1");
//    for (int i = 0; i < pointsNum; i++) {
//        fprintf(plotter, "%lf %lf\n", killers[i], victims[i]);
//    }
//    fprintf(plotter, "%c\n", 'e');

#ifdef WIN64
    _pclose(plotter);
#else
    pclose(plotter);
#endif
```

```
    return 0;
}
```

cout << Lotka-Volterra<< endl;

1

2

1

0.3

0.5

0.3

50

50

Figure 1.

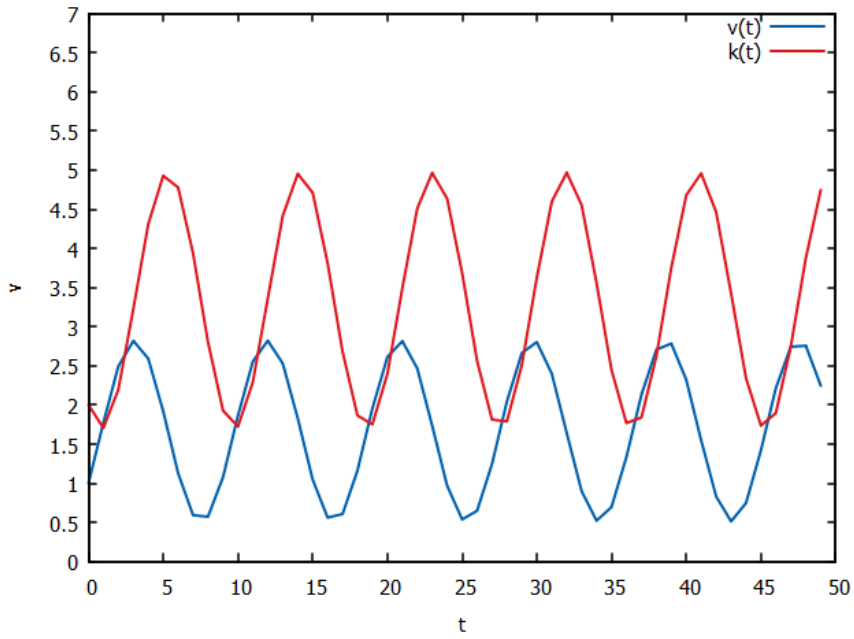

Figure 2.