Analytical Geometry and Linear Algebra II

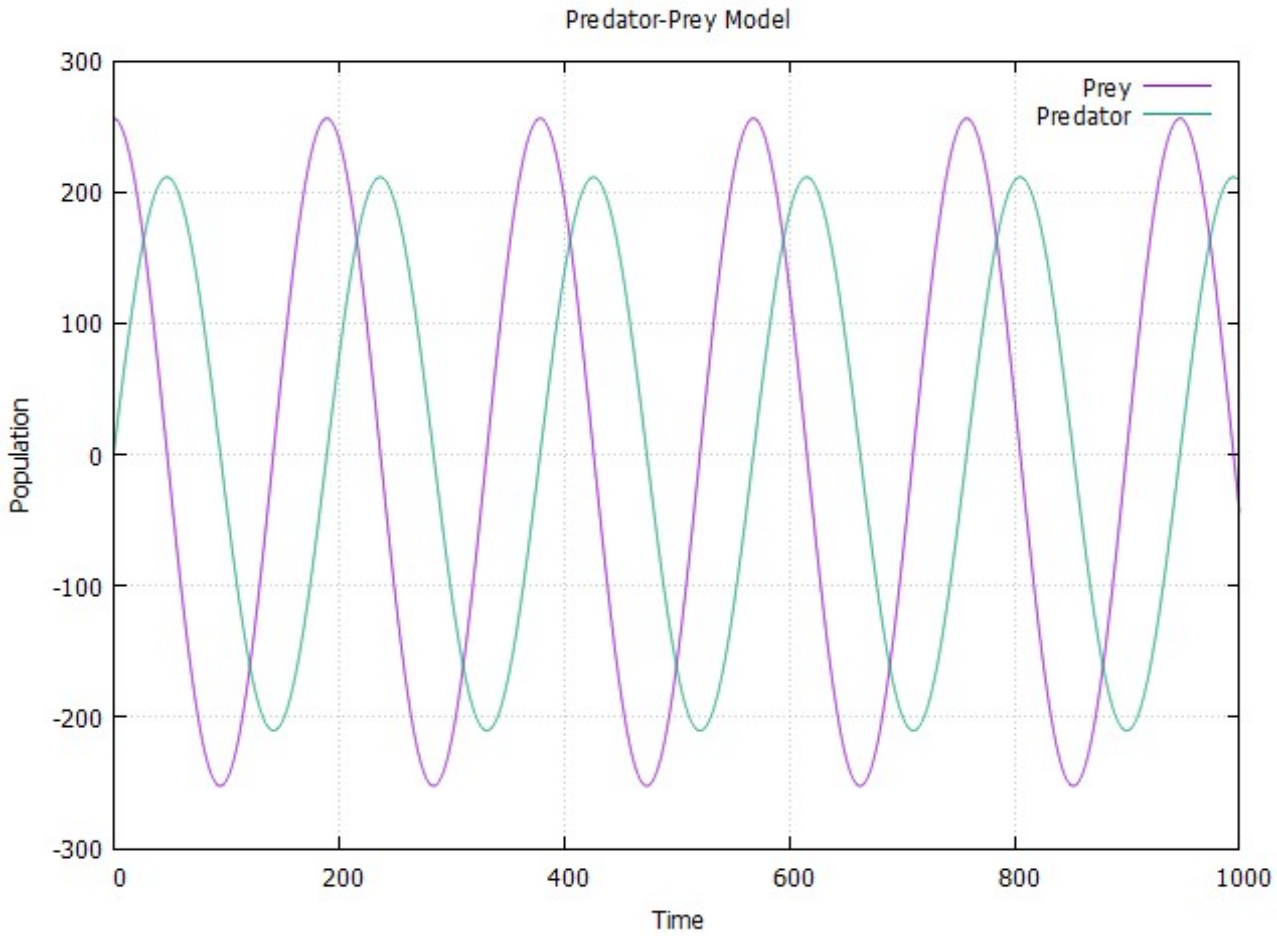
Dmitrii Kuzmin

April, 2023

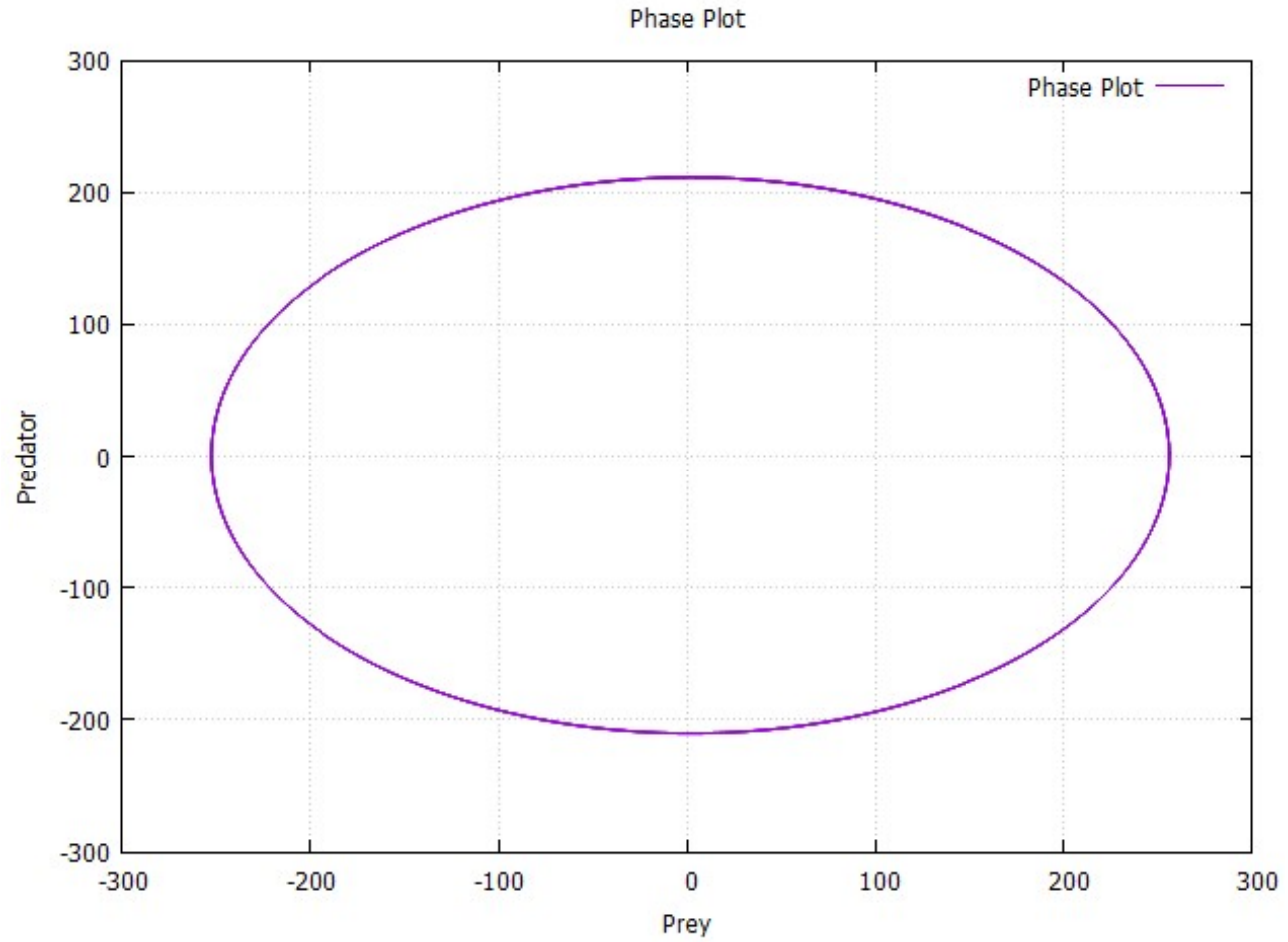
DSAI-03
dm.kuzmin@innopolis.university

GitHub link: https://github.com/1kkiRen/AGLAJoint3

Plot x(t) and k(t):



Plot x(k):



Experimental data:

initial v	initial k	α_1	β_1	α_2	β_2	time	number of the points of approximation
256	1	0.011	0.022	0.1	0.055	1000	10000

Source code (with gnuplot):

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

using namespace std;

#ifdef WIN32
    #define GNUPLOT_NAME "C:\\gnuplot\\bin\\gnuplot -persist"
#else
    #define GNUPLOT_NAME "gnuplot -persist"
#endif
int main() {
    #ifdef WIN32
        FILE* pipe = _popen(GNUPLOT_NAME, "w");
        FILE* pipe1 = _popen(GNUPLOT_NAME, "w");
    #else
        FILE* pipe = popen(GNUPLOT_NAME, "w");
```

```
    FILE* pipe1 = popen(GNUPLOT_NAME, "w");
#endif

double V0, K0;
cin >> V0 >> K0;

double alpha1, beta1, alpha2, beta2;
cin >> alpha1 >> beta1 >> alpha2 >> beta2;

double time;
cin >> time;

double points;
cin >> points;

double step = time / points;

double _V0 = (alpha2 / beta2);
double _K0 = (alpha1 / beta1);

double v0 = V0 - _V0;
double k0 = K0 - _K0;

vector<double> Vs;
vector<double> Ks;

for (double i = 0; i <= time; i += step){
    Vs.push_back(_V0 + v0 * cos(sqrt(alpha1 * alpha2) * i) - k0 * ((sqrt(alpha2) * beta1) / (beta2
        * sqrt(alpha1))) * sin(sqrt(alpha1 * alpha2) * i));
    Ks.push_back(_K0 + v0 * ((sqrt(alpha1) * beta2) / (beta1 * sqrt(alpha2))) * sin(sqrt(alpha1 *
        alpha2) * i) + k0 * cos(sqrt(alpha1 * alpha2) * i));
}

fprintf(pipe, "set title 'Predator-Prey Model'\n");
fprintf(pipe, "set xlabel 'Time'\n");
fprintf(pipe, "set ylabel 'Population'\n");
fprintf(pipe, "set grid\n");
fprintf(pipe, "plot '-' with lines title 'Prey', '-' with lines title 'Predator'\n");

for (int i = 0; i < Vs.size(); i++){
    fprintf(pipe, "%lf %lf\n", i * step, Vs[i]);
}
fprintf(pipe, "e\n");

for (int i = 0; i < Ks.size(); i++){
    fprintf(pipe, "%lf %lf\n", i * step, Ks[i]);
}
fprintf(pipe, "e\n");

fflush(pipe);

fprintf(pipe1, "set title 'Phase Plot'\n");
fprintf(pipe1, "set xlabel 'Prey'\n");
fprintf(pipe1, "set ylabel 'Predator'\n");
fprintf(pipe1, "set grid\n");
fprintf(pipe1, "plot '-' with lines title 'Phase Plot'\n");

for (int i = 0; i < Vs.size(); i++){
    fprintf(pipe1, "%lf %lf\n", Vs[i], Ks[i]);
}


fflush(pipe1);

#ifdef WIN32
    _pclose(pipe);
```

```
        _pclose(pipe1);
    #else
        pclose(pipe);
        pclose(pipe1);
    #endif
    return 0;
}
```

Source code (without gnuplot):

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

using namespace std;


int main() {
    cout << fixed << setprecision(2);

    double V0, K0;
    cin >> V0 >> K0;

    double alpha1, beta1, alpha2, beta2;
    cin >> alpha1 >> beta1 >> alpha2 >> beta2;

    double time;
    cin >> time;

    double points;
    cin >> points;

    double step = time / points;

   double _V0 = (alpha2 / beta2);
    double _K0 = (alpha1 / beta1);

    double v0 = V0 - _V0;
    double k0 = K0 - _K0;

    vector<double> Vs;
    vector<double> Ks;

    cout << "t:\n";

    for (double i = 0; i <= time; i += step){
        cout << i << " ";
        Vs.push_back(_V0 + v0 * cos(sqrt(alpha1 * alpha2) * i) - k0 * ((sqrt(alpha2) * beta1) / (beta2
            * sqrt(alpha1))) * sin(sqrt(alpha1 * alpha2) * i));
        Ks.push_back(_K0 + v0 * ((sqrt(alpha1) * beta2) / (beta1 * sqrt(alpha2))) * sin(sqrt(alpha1 *
            alpha2) * i) + k0 * cos(sqrt(alpha1 * alpha2) * i));
    }

    cout << "\nv:\n";
    for (int i = 0; i < Vs.size(); i++){
        cout << Vs[i] << " ";
    }

    cout << "\nk:\n";
    for (int i = 0; i < Ks.size(); i++){
        cout << Ks[i] << " ";
    }
```

```
    return 0;
}
```