Predator-Prey Model

Izmailov Ruslan DS-02
r.izmailov@innopolis.university

April 2023

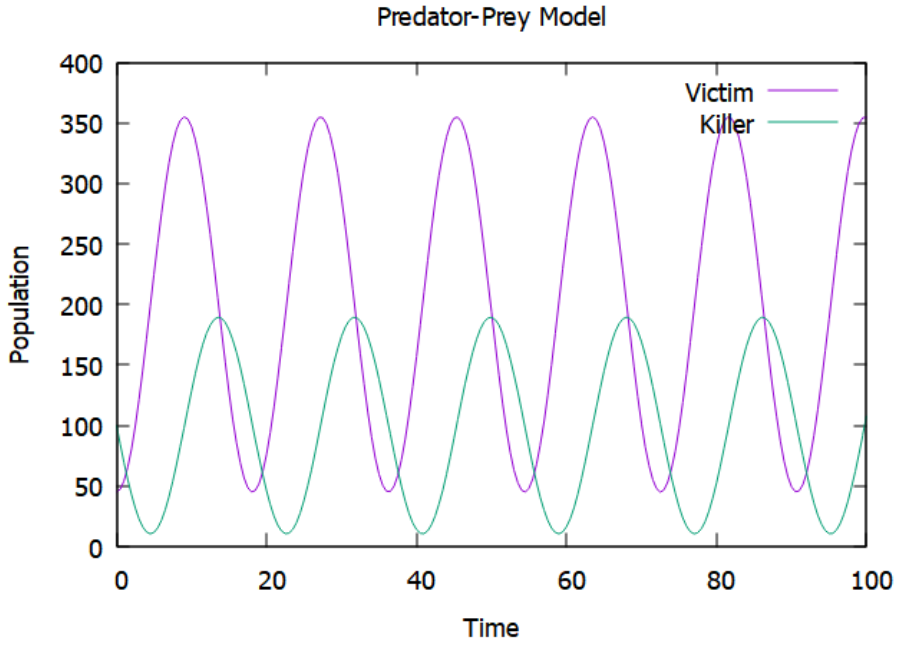
The code of the task, input data and plots available in the github perository:
https://github.com/Hexy00123/PredatorPreyModel/tree/main

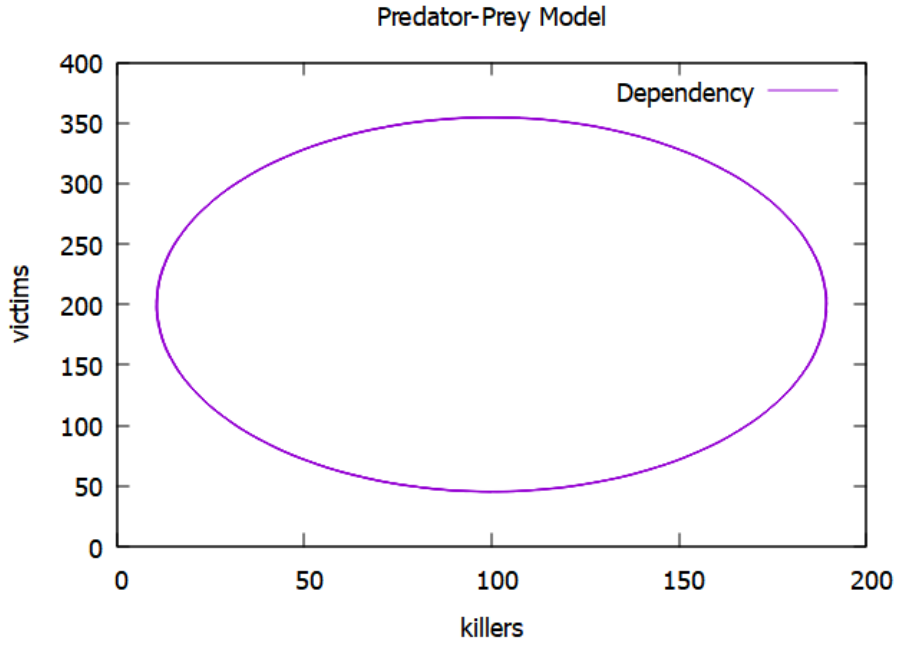
To test this model I used
45 victims and 100 killers at the beginning
$\alpha_1 = 0.3$
$\beta_1 = 0.003$
$\alpha_2 = 0.4$
$\beta_2 = 0.002$
Time of simulation = 100
Number of time points = 500

Therefore, input looks like this:

```
45
100
0.3
0.003
0.4
0.002
100
200
```

The resulting plots are follows:


Predator-Prey Model
Victim
Killer
Population
Time
0
50
100
150
200
250
300
350
400
0
20
40
60
80
100


Predator-Prey Model
Dependency
victims
killers
0
50
100
150
200
250
300
350
400
0
50
100
150
200

Source code from yandex contest:

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

using namespace std;

void predatorPrey(int vs, int ks, double a1, double a2, double b1, double b2, int timeLimit, int approximation) {
    /*
     * v'(t) = alpha1 * v(t) + beta1 * v(t) * k(t),
     * k'(t) = -alpha2 * v(t) + beta2 * v(t) * k(t)
     */

    auto time = vector<double>();
    for (double i = 0; i <= timeLimit; i += (double) timeLimit / (double) approximation) {
        time.push_back(i);
    }

    double v0 = vs - a2 / b2;
    double k0 = ks - a1 / b1;

    auto v = vector<double>();
    auto k = vector<double>();
    for (auto t: time) {
        v.push_back(
                v0 * cos(sqrt(a1 * a2) * t) - k0 * sqrt(a2) * b1 * sin(sqrt(a1 * a2) * t) / (b2 * sqrt(a1)) + a2 / b2);

        k.push_back(
                v0 * sqrt(a1) * b2 * sin(sqrt(a1 * a2) * t) / (b1 * sqrt(a2)) + k0 * cos(sqrt(a1 * a2) * t) + a1 / b1);
    }


    cout << "t:\n";
    for (auto value: time)
        cout << value << " ";
    cout << "\n";

    cout << "v:\n";
    for (auto value: v)
        cout << value << " ";
    cout << "\n";

    cout << "k:\n";
    for (auto value: k)
        cout << value << " ";
}


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

```
52
53      int numberOfVictims, numberOfKillers;
54      cin >> numberOfVictims >> numberOfKillers;
55
56      double a1, a2, b1, b2;
57      cin >> a1 >> b1 >> a2 >> b2;
58
59      int timeLimit, approximationPoints;
60      cin >> timeLimit >> approximationPoints;
61
62      predatorPrey(numberOfVictims, numberOfKillers, a1, a2, b1, b2,
        timeLimit, approximationPoints);
63      return 0;
64  }
```

Source code for plotting:

```
1   #include <iostream>
2   #include <vector>
3   #include <cmath>
4   #include <iomanip>
5
6   using namespace std;
7
8   vector<vector<double>>
9   predatorPrey(int vs, int ks, double a1, double a2, double b1,
        double b2, int timeLimit, int approximation) {
10      /*
11       * v'(t) = alpha1 * v(t) + beta1 * v(t) * k(t),
12       * k'(t) = -alpha2 * v(t) + beta2 * v(t) * k(t)
13       */
14
15      auto time = vector<double>();
16      for (double i = 0; i <= timeLimit; i += (double) timeLimit / (
        double) approximation) {
17          time.push_back(i);
18      }
19
20      double v0 = vs - a2 / b2;
21      double k0 = ks - a1 / b1;
22
23      auto v = vector<double>();
24      auto k = vector<double>();
25      for (auto t: time) {
26          v.push_back(
27                  v0 * cos(sqrt(a1 * a2) * t) - k0 * sqrt(a2) * b1 *
        sin(sqrt(a1 * a2) * t) / (b2 * sqrt(a1)) + a2 / b2);
28
29          k.push_back(
30                  v0 * sqrt(a1) * b2 * sin(sqrt(a1 * a2) * t) / (b1 *
         sqrt(a2)) + k0 * cos(sqrt(a1 * a2) * t) + a1 / b1);
31      }
32
33      auto res = vector<vector<double>>();
34      res.push_back(time);
35      res.push_back(v);
36      res.push_back(k);
```

```
    return res;
}

#ifdef WIN32
#define GNUPLOT_NAME "C:\\gnuplot\\bin\\gnuplot -persist"
#else
#define GNUPLOT_NAME "gnuplot -persist"
#endif

int main() {
#ifdef WIN32
    FILE *victimsAndKillersOfTime = _popen(GNUPLOT_NAME, "w");
    FILE *victimsOfKillers = _popen(GNUPLOT_NAME, "w");
#else
    FILE* victimsAndKillersOfTime = popen(GNUPLOT_NAME, "w");
    FILE *victimsOfKillers = _popen(GNUPLOT_NAME, "w");
#endif

    cout << fixed;
    cout << setprecision(2);

    int numberOfVictims, numberOfKillers;
    cin >> numberOfVictims >> numberOfKillers;

    double a1, a2, b1, b2;
    cin >> a1 >> b1 >> a2 >> b2;

    int timeLimit, approximationPoints;
    cin >> timeLimit >> approximationPoints;

    auto res = predatorPrey(numberOfVictims, numberOfKillers, a1,
    a2, b1, b2, timeLimit, approximationPoints);
    auto t = res[0], v = res[1], k = res[2];

    fprintf(victimsAndKillersOfTime, "set title 'Predator-Prey
    Model'\n");
    fprintf(victimsAndKillersOfTime, "set xlabel 'Time'\n");
    fprintf(victimsAndKillersOfTime, "set ylabel 'Population'\n");
    fprintf(victimsAndKillersOfTime,
            "plot '-' using 1:2 with lines title 'Victim', '-'
    using 1:2 with lines title 'Killer'\n");

    fprintf(victimsOfKillers, "set title 'Predator-Prey Model'\n");
    fprintf(victimsOfKillers, "set xlabel 'killers'\n");
    fprintf(victimsOfKillers, "set ylabel 'victims'\n");
    fprintf(victimsOfKillers,
            "plot '-' using 1:2 with lines title 'Dependency', '-'
    using 1:2 with lines title 'Predator'\n");

    // Plot the prey and predator populations
    for (int i = 0; i < t.size(); ++i)
        fprintf(victimsAndKillersOfTime, "%f\t%f\n", t[i], v[i]);
    fprintf(victimsAndKillersOfTime, "e\n");

    for (int i = 0; i < t.size(); ++i)
        fprintf(victimsAndKillersOfTime, "%f\t%f\n", t[i], k[i]);
    fprintf(victimsAndKillersOfTime, "e\n");
```

```
90
91      for (int i = 0; i < t.size(); ++i) {
92          fprintf(victimsOfKillers, "%f\t%f\n", k[i], v[i]);
93      }
94      fprintf(victimsOfKillers, "e\n");
95
96      fflush(victimsAndKillersOfTime);
97      fflush(victimsOfKillers);
98  #ifdef WIN32
99      _pclose(victimsAndKillersOfTime);
100     _pclose(victimsOfKillers);
101 #else
102     pclose(victimsAndKillersOfTime);
103     pclose(victimsOfKillers);
104 #endif
105
106     return 0;
107 }
```