Számítógépes szimulációk labor - Populációdinamika

Zsigmond István, iiqcc8

Április 27, 2019

1. Bevezetés

1.1. Populációdinamika

Newton eredetileg a fizikai rendszerek jobb leírása végett vezette be a differenciálegyenleteket, de hamar kiderült, hogy ez is, mint a matematikai sok más eleme elég szépen leírja a biológiai természet egyes mintáit és működését, illetve tökéletesen alkalmas teljesen más (pl. biztonsági, pénzügyi) "rendszerek" leírásáa is. Természetesen ezek nem-lineáris rendszerek lineárisan való közelítése, ami azt jelenti, hogy habár kellően reprezentatívak, a természet motívumait, bizonyos szituációkat nem lehet velük 100%-os pontossággal leírni.

A populációdinamika egy olyan leíró rendszer, mely - ahogy a neve is sejteti - egy valamennyi (n) elemből álló rendszer időbeli változását tárgyalja az elemszám szempontjából.

1.2. Matematikai leírás

1.2.1. Egyszerű születési-halálozási rendszer

Ha csak szaporodási rátát veszünk figyelembe (az a mennyiség, mely a populáció időegységenként való növekedésének számát határozza meg), a populáció elemszámának változása $\mathbf{\Delta t}$ időközönként:

$$n(t + \Delta t) = n(t) + an(t) \tag{1}$$

ahol **a** a szaporodási ráta. Ezt átrendezve és határértéket vizsgálva:

$$\frac{dn(t)}{dt} = lim_{\Delta t \to 0} \frac{n(t + \Delta t) - n(t)}{\Delta t} = an(t) \tag{2}$$

differencálegyenletet kapjuk eredményként. Ez a megoldás akkor jó közelítés, ha a populáció mérete nagy. a 2. egyenlet egy szeparábilis differenciálegyenlet, melyet egyszerűen meglehet oldani.

$$\frac{dn}{n} = adt$$

kiintegrálva mind a két oldalt kapjuk, hogy

$$\int \frac{dn}{n} = ln(n) = \int adt = at + C$$

Az integrációs konstanst nullának választva ($\mathbf{C = 0}$), a két oldalt exponenciálisra emelve kapjuk meg a $\mathbf{n(t)}$ általános megoldását:

$$n(t) = e^{at} \tag{3}$$

Tovább bonyolítva a rendszert valósághűbb, ha bevezetünk a szaporodási ráta mellé egy elmúlási/halálozási rátát is, mely annak felel meg, hogy időnként a populáció 1-1 eleme eltűnik a rendszerből a beérkezések mellett. Ekkor a differenciálegyenletünk a következő alakban módosul:

$$\frac{dn}{dt} = an - dn \tag{4}$$

ahol **d** a halálozási ráta. Ennek megoldása:

$$n(t) = e^{rt} \tag{5}$$

alakú, ahol $\mathbf{r = a - d}$. $\mathbf{r}$ előjelétől függően ez egy exponenciálisan növekedő vagy csökkenő rendszer lesz. Továbbbonyolítva olyan paraméterrel, mely figyelembe veszi az élelem és/vagy túléléshez szökséges nyersanyag korlátoltságát, a differenciálegyenletünk a következő alakot ölti:

$$\frac{dn}{dt} = rnF(n) \tag{6}$$

ahol $\mathbf{F(n)}$ jelöli a erőforrások méretét a populáció pillanatnyi elemszámának függvényében. Érezhető, hogy egy ilyen paraméter bevezetése nagyon könnyen nagyon hamar eltud bonyolódni - még-realisztikusabb esetekben ez a paraméter az időnek is a függvénye. $\mathbf{F(n)}$ egy legegyszerűbb, $\mathbf{t}$-től független és $\mathbf{n}$-ben lineáris alakja egy olyan paraméter, amely a populáció elemszámának változásával arányosan, ugyanabban a pillanatban nő vagy csökken. Ennek tudni kell azt, hogy $\mathbf{n = 0}$ esetben az erőforrások száma maximális ($\mathbf{F(n = 0) = 1}$), illetve egy, a populáció elemszámára vett felső határnál ne engedjen további szaporulást ($\mathbf{F(n = k) = 0}$). Ebből megkonstruálhatjuk $\mathbf{F(n)}$ alakját:

$$F(n) = 1 - \frac{n}{k} \tag{7}$$

mely a fenti feltételeket kielégíti.

Ekkora a differenciálegyenletünk a következő alakban módosul:

$$\frac{dn}{dt} = rn(1 - \frac{n}{k}) \tag{8}$$

Új változó bevezetésével ($\mathbf{x = \frac{n}{k}}$) átskálázva a differenciálegyenlet:

$$\frac{dx}{dt} = rx(1 - x) \tag{9}$$

Ennek megoldásként kapjuk meg az ún. logisztikus egyenletet:

$$x(t) = \frac{1}{1 + (\frac{1}{x_0} - 1)e^{-rt}} \tag{10}$$

mely egy, a kezdeti $\mathbf{r}$ illetve $\mathbf{x_0}$ paraméterektől függő, növekedő vagy csökkenő szigmoid jellegű görbét ír le.

1.2.2. Fixpontok vizsgálata

A logisztikus egyenlet fixpontjai (ahol $\mathbf{\frac{dx}{dt} = 0}$) jelen esetben az $\mathbf{x = 0}$ és az $\mathbf{x = 1}$. A fixpontok lehetnek stabilak vagy instabilak, melyet a rendszer kis kitérítésével lehet vizsgálni - ha stabil, visszatér a kezdeti pontba, ha instabil, akkor elszalad (jelen esetben exponenciálisan). A stabilitást lineáris perturbációval a legegyszerűbb vizsgálni; vegyünk egy differenciálegyenletet a következő alakban:

$$\frac{dx}{dt} = f(x) \tag{11}$$

Legyen a fixpont $\mathbf{x^*}$, $\mathbf{f(x^*) = 0}$, kis perturbáció pedig $\epsilon(\mathbf{t})$. Ekkor a kitérés:

$$x(t) = x^* + \epsilon(t) \tag{12}$$

Ezt beírva a 11. egyenletbe és Taylor-sorba fejtve:

$$\frac{dx}{dt} = \frac{dx^*}{dt} + \frac{d\epsilon}{dt} = \frac{d\epsilon}{dt} = f(x^* + \epsilon) = f(x^*) + \epsilon f'(x^*) + ... \tag{13}$$

Elhagyva a magasabb rendű deriváltakat és felhasználva, hogy $\mathbf{f(x = x^*) = 0}$, a differenciálegyenletünk a következő alakot ölti:

$$\frac{d\epsilon}{dt} = \epsilon f'(x^*) \tag{14}$$

melynek megoldása:

$$\epsilon(t) = \epsilon(0)e^{f'(x*)t}. \tag{15}$$

Ez a megoldás akkor stabil, ha $\mathbf{\lim_{t\to 0}}\epsilon(\mathbf{t}) = \mathbf{0}$, tehát amikor $\mathbf{f'(x^*) < 0}$.

1.2.3. Populációk versengése közös erőforrásért

Vegyünk két populációt $(\mathbf{n_1}, \mathbf{n_2})$, melyek közös erőforrásért küzdenek. Legegyszerűbb esetben ez azt jelenti, hogy ha az egyik populáció szaporodik, a másik faj számára kevesebb erőforrás áll rendelkezésre a szaporodáshoz. Ekkor két populáció elemszámának időbeli változását leíró differenciálegyenletek:

$$\frac{dn_1}{dt} = r_1 n_1 \left(1 - \frac{n_1 + \alpha n_2}{k_1}\right) \tag{16}$$

$$\frac{dn_2}{dt} = r_2 n_2 \left(1 - \frac{n_2 + \alpha n_1}{k_2}\right) \tag{17}$$

ahol α, β paraméterek azt fejezik ki, hogy milyen arányban fogyasztja az egyik faj a másik erőforrásait. A rendszer fixpontjai így mostmár $\alpha, \beta, \mathbf{r_1}, \mathbf{r_2}, \mathbf{k_1}, \mathbf{k_2}$ függvénye.

1.2.4. Ragadozó-préda rendszerek

A populációk kölcsönhatása itt nem csak az erőforrások egymástól való elvételében nyilvánul meg, hanem abban is, hogy az egyik, dominánsabb populáció erőforrásai között tudja a másik populációt is. Erre példa egy "nyúl-róka" rendszer, ahol kezdetben sok nyúl van, ez sok táplálékot jelent a rókáknak, melyek így elszaporodnak. Egyre több nyúl fogy, tehát a nyúlak populációjának elemszáma csökken - a rókák nem tudnak szaporodni. Ha a rókák nem szaporodnak, a nyulak létszáma megnő, ezzel együtt a rókák is újra eltudnak kezdeni szaporodni, és ez a folyamat megy végbe újra és újra. Egy ilyen rendszer leírására alkalmas a Lotka-Volterra-modell, mely korlátlan téplálékforrást feltételez a nyulaknak (R) és korlátlan kapacitást enged a rókáknak (F), mely utóbbi az első feltételből következik. Így a populáció időbeli változását leíró differenciálegyenletek:

$$\frac{dn_R}{dt} = an_R - bn_F n_R \tag{18}$$

$$\frac{dn_F}{dt} = Cn_F n_R - dn_F \tag{19}$$

ahol

- $\mathbf{n_R}$ a nyulak száma,
- $\mathbf{n_F}$ a rókák száma,
- $\mathbf{a}$ a nyulak szaporodási rátája
- $\mathbf{cn_R}$ a rókák szaporodási rátája
- $\mathbf{bn_F}$ a nyulak halálozási rátája
- $\mathbf{d}$ a rókák halálozási rátája

Ez a modell valósszerűbbé tehető, ha korlátozzuk a nyulak erőforrásait $(\mathbf{a} \rightarrow \mathbf{a}(\mathbf{1} - \frac{\mathbf{n_R}}{\mathbf{k}}))$, illetve korlátozzuk a rókák nyúlfogyasztási képességét $(\mathbf{n_R n_F} \rightarrow \frac{\mathbf{n_R n_F}}{\mathbf{1+n_R s}})$.

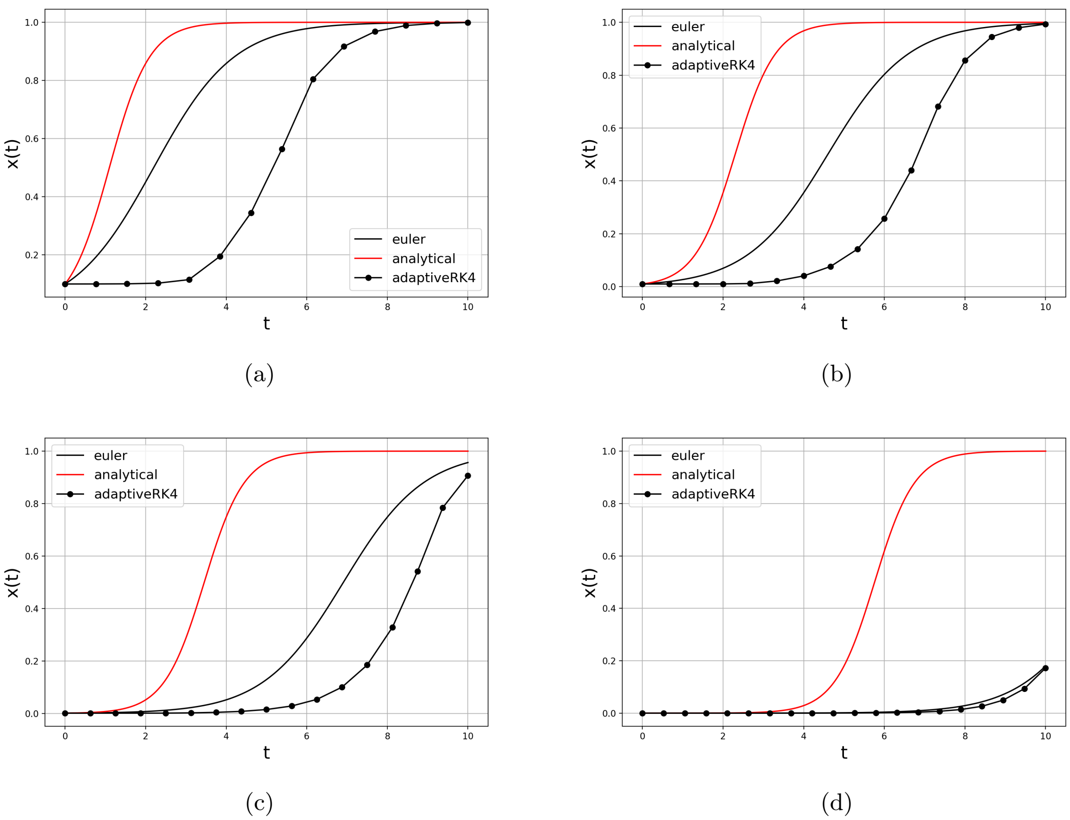
2. Feladatok megvalósítása

2.1. Logisztikus egyenlet fixpont körüli viselkedésének vizsgálata

Az első feladat során a rendszer fixpontok közelében és azoktól távoli pontokban való viselkedését szeretnénk vizsgálni a rendszernek. Az a várakozás, hogy a fixpontoktól nagyon kicsit elmozdulva a rendszer még stabil marad és nem fut el exponenciálisan. A feladat további részét képezte annak a vizsgálata, hogy a korábban már alkalmazott Euler-módszer és adaptív Runge-Kutta múdszer milyen jól adja vissza az analitikus megoldás eredményét.

Első lépésben megírtam a szimulációhoz szükséges kódot (popdyn.cpp), melyben az adaptív módszer implementálásához segítségemre volt egy korábban használt, bolygómozgást szimuláló példakód [2]. A futtatások eredményeit Python-ban értékeltem ki.

Elindítva a programot az néhány szükséges információ bekérése után visszaad egy adatfájlt, melyet megfelelően kiértékelve jutottam az 1. és a 2. ábrákhoz. az 1. ábra mutatja a rendszer fixpont körüli viselkedését, a fixpont stabilitását.

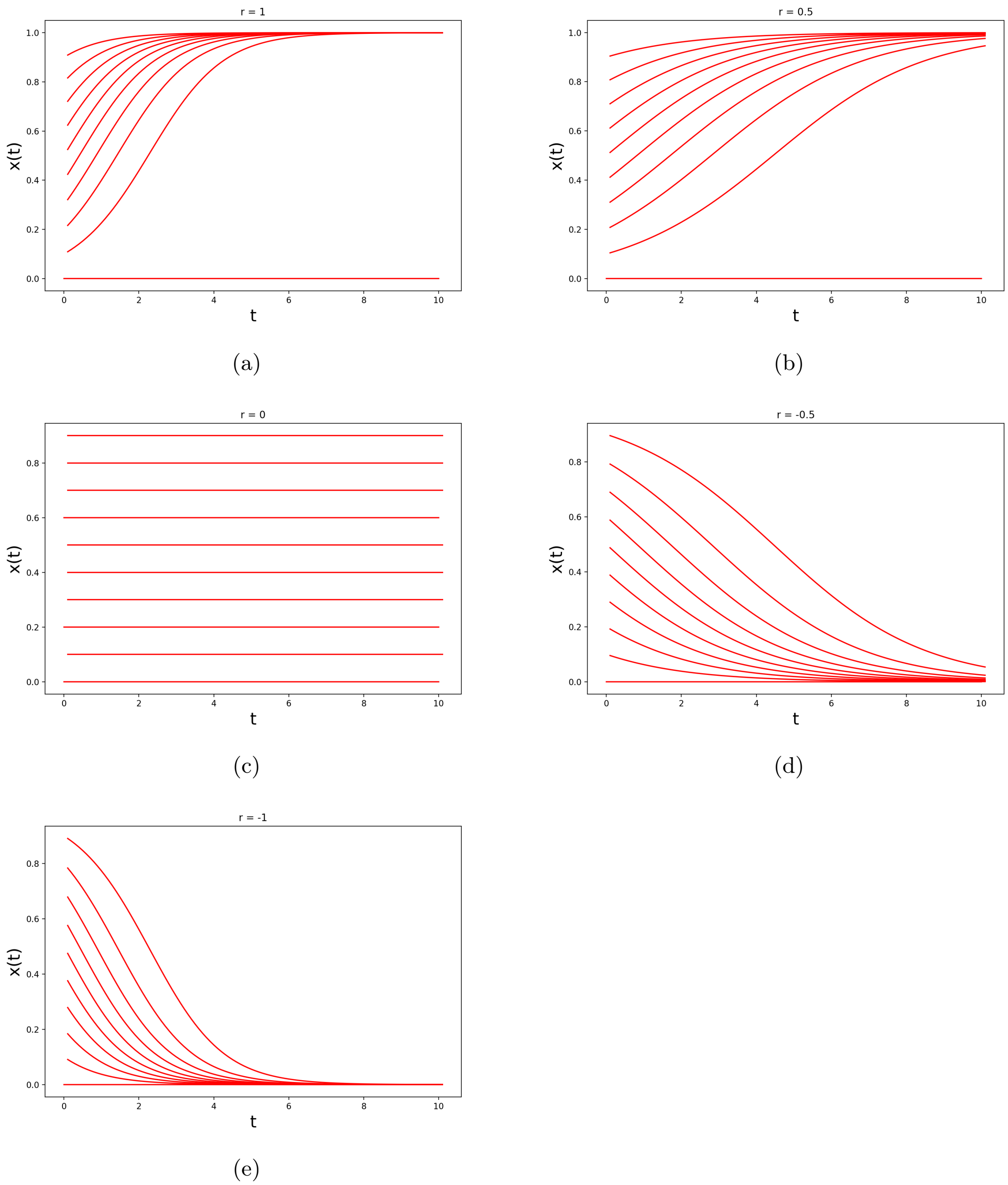


(a) (b) (c) (d)

1. ábra. Telítettség időfejlődése $\mathbf{r = 1}$ és különböző $\mathbf{x_0}$ kezdeti telítettség esetén. (a) $\mathbf{x_0 = 0.1}$, (b) $\mathbf{x_0 = 0.01}$, (c) $\mathbf{x_0 = 0.001}$, (d) $\mathbf{x_0 = 0.00001}$. Látható, hogy ahogy csökkentjük a telítettség kezdeti értékét, úgy egyre jobban fekszenek egymásra az Euler-módszer és az Adaptív Runge-Kutta módszer eredményei. A kezdeti értékek változtatásából látszik, mennyire stabil a rendszer - ha a fixpont nagyon kicsi ($\mathbf{x_0 \ll 1}$), a rendszer hosszú ideig nem fut el az egyensúlyi ponttól.

Érdekes látni, ahogy az instabil pont körül a két módszer csak nagyon lassan szakad el, míg az analitikus módszer látszólag sokkal kevésbé érzékeny. Távolabb menve az instabil ponttól azonban mind a két módszer elkezdi megközelíteni az analitikus megoldást, melyek közül az Euler-módszer gyorsabban teszi ezt. Ennek okán választottam a következő alfeladat kiértékeléséhez az Euler-módszer által szolgáltatott eredményeket.

A következő alfeladatban a különböző kezdő- és r paraméterek mellett kellett kirajzolni a logisztikus egyenletet, ezzel reprodukálva a feladatkiírás idetartozó ábráit. Ennek eredményeit mutatja a 2. ábra.

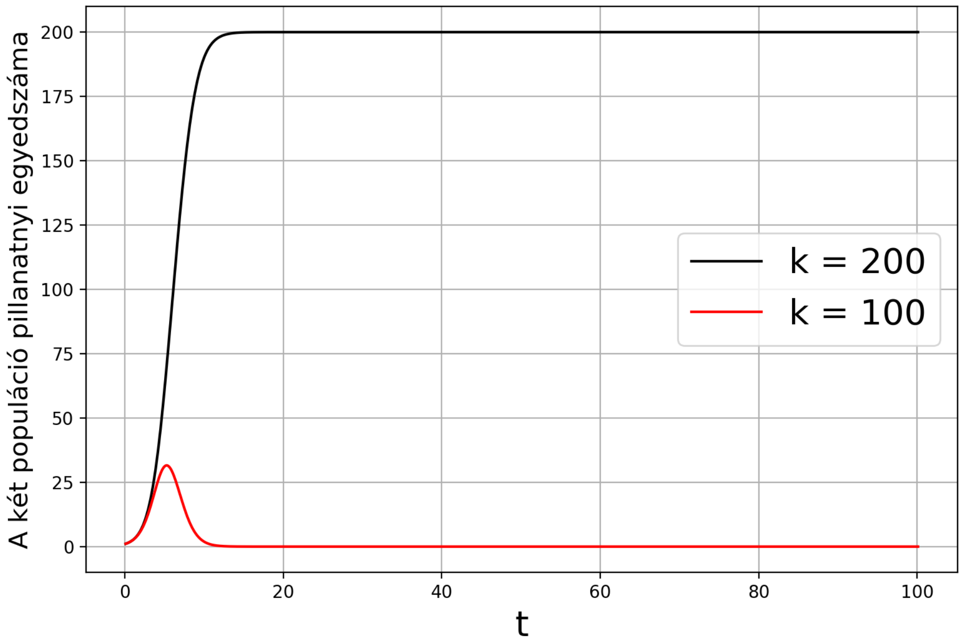


2. ábra. Telítettség időfejlődése különböző $\mathbf{r}$ esetén. (a) $\mathbf{r = 1}$, (b) $\mathbf{r = 0.5}$, (c) $\mathbf{r = 0}$, (d) $\mathbf{r = -0.5}$, (e) $\mathbf{r = -1}$. Az ábrákhoz az Euler-módszer numerikus eredményeit alkalmaztam.

2.2. Fajok közös erőforrásért való versengése

A második feladatban két faj közös erőforráson való versengésének szimulálása volt a kitűzött feladat. Ehhez első feladathoz írt kódomat módosítottam a 16. és a 17. egyenleteken alapuló Euler-algoritmus bevezetésével (popdyn multipop.cpp). a 3. ábra szemlélteti az így kapott eredményeket.

Nagyon érdekes a szimuláció eredménye: látszólag a nagyobb $\mathbf{k}$ egyedszám-limittel rendelkező faj dominanciája nem csak abban nyilvánul meg, hogy limitálja a másikat, de nagyon hamar kihalásra is ítéli azt, lecsökkentve annak egyedszámát 0-ra és 0-n tartva. Innen látszik az, hogy $\alpha = \beta = \mathbf{1}$ esetén valóban nem létezhet stabilan együtt.



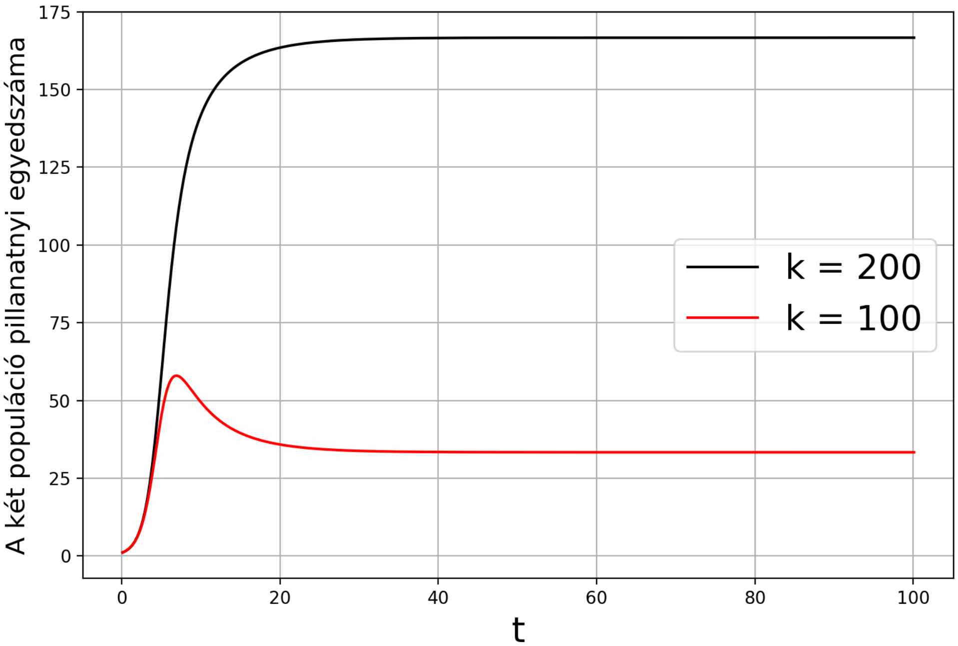
3. ábra. Fajok versengése közös erőforrásokért, $\alpha = \beta = \mathbf{1}$. A szimuláció során a nagyobb $\mathbf{k}$ értékkel indított populáció nőtt ki dominánsként.

További része ennek a feladatnak megmutatni, hogy két faj akkor maradhat meg egymás mellett stabilan, ha $\alpha\mathbf{k_2} < \mathbf{k_1}$, illetve $\beta\mathbf{k_1} < \mathbf{k_2}$. Egy ilyen megválasztás lehet a korábbi $\mathbf{k_1} = \mathbf{200}$ és $\mathbf{k_2} = \mathbf{100}$ értékek mellett az $\alpha = \mathbf{1}$, $\beta = \mathbf{0.4}$. Ennek eredményét szemlélteti a 4. ábra.

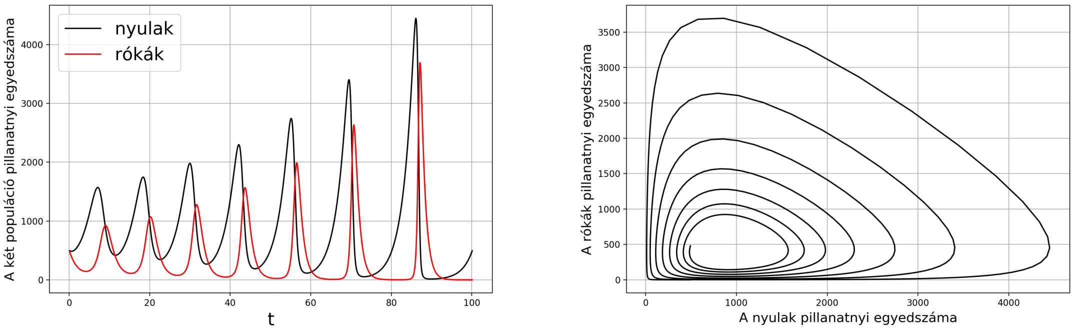
Láthatjuk, hogy olyan paraméterek mellett elvégezve a szimulációt, melyek kielégítik a stabilitásra felírt feltételt, valóban azt eredményezték, hogy a két faj stabilan megtud élni közös erőforráson osztozva.

2.3. Lotka-Volterra-modell implementációja

Az utolsó feladat célja a Lotka-Volterra-modell implementációja és a hozzá tartozó egyszerű példának, a róka-nyúl rendszer szimulációja. Ennek megoldásához a második feladathoz írt kódomat átírtam úgy, hogy a 18. és a 19. egyenleteken alapúló Euler-algoritmust kezeljen, illetve megváltoztattam a szükséges paramétereket (poldyn LV.cpp). A szimuláció eredményét az 5. ábra mutatja. A felhasznált paraméter a fólián bemutatott példák egyikéből származnak. Sajnos az így kapott eredményem nem tűkröző a fólián demonstráltakat.



4. ábra. Fajok versengése közös erőforrásokért, $\alpha = \mathbf{1}$ $\beta = \mathbf{0.4}$. A szimuláció során továbbra is a nagyobb **k** értékkel indított populáció nőtt ki dominánsként, de ezúttal a két faj stabilan tud létezni egymás mellett.



(a) (b)

5. ábra. Lotka-Volterra modell "róka-nyúl" példájának időfejlődése, a nyulak számára korlátlan táplálékot, a rókák számára korlátlan kapacitást feltételezve. Kezdetben 500-500 egyed képezi mindkét populációt. (a) $\mathbf{a = 0.4}$, (b) $\mathbf{b = 0.001}$, (c) $\mathbf{c = 0.001}$, (d) $\mathbf{d = 0.9}$. A felhasznált paraméterek a feladatkiírás fóliáján bemutatott példákból vannak. Érdekes, hogy habár nem vezettem be extra feltételeket telítettségre, mégis inkább egy olyan esethez hasonló eredményt kaptam a szimulációból.

Függelék

Az első feladat megvalósításához írt kód:

```
#include <cmath>
#include <cstdlib>
#include <fstream>
#include <iostream>
using namespace std;

#include "vector.hpp"
#include "odeint.hpp"
using namespace cpl;

Vector f(const Vector& x) {

    double t = x[0], Xa = x[1], r = 1;
    Vector f(2);
    f[0] = 1;
    f[1] = r * Xa * (1 - Xa);

return f;
}

int main() {
   double r = 1, X0 = 0.1, dx, X, tmax, dt, t, accuracy;

   cout << "Enter the time limit, time’s starting point and step size: ";
   cin >> tmax >> t >> dt;
   cout << "Accuracy for adaptive method: ";
   cin >> accuracy;

   clock_t tStart = clock();

   ofstream dataFile("euler.data");

   dx = dt * r * X0 * (1 - X0);
   X = X0 / (X0 + (1 - X0) * exp(-r * t));

   while (t < tmax || X0 >= 1) {

       t += dt;
       X0 += dx;

       dx = dt * r * X0 * (1 - X0);

       X = X0 / (X0 + (1 - X0) * exp(-r * t));

       dataFile << t << ’ ’ << X0 << ’ ’ << X << ’\n’;

   }

   dataFile.close();

   cerr << "Time elapsed for Euler method: " << (double)(clock() - tStart)/CLOCKS_PER_SEC
       << " s" << endl;

   clock_t tStart2 = clock();
```

```
    dataFile.open("adaptive.data");

    Vector x0(2);
    x0[0] = 0, x0[1] = 0.1;

    Vector x = x0;

    do {
        for (int i = 0; i < 2; i++) {
            dataFile << x[i] << '\t';
        }
        dataFile << '\n';

        adaptiveRK4Step(x, dt, accuracy, f);

    } while (x[0] < tmax || x[1] >= 1);

    dataFile.close();

    cerr << "Time elapsed for adaptiveRK4 method: " << (double)(clock() -
        tStart2)/CLOCKS_PER_SEC << " s" << endl;
}
```

A második feladat megvalósításához írt kód:

```
#include <cmath>
#include <cstdlib>
#include <fstream>
#include <iostream>
using namespace std;


int main() {
    double r1 = 1, r2 = 1, n1 = 1, n2 = 1, k1 = 200, k2 = 100, dn1, dn2, tmax, dt, t,
        alpha, beta;

    cout << "Enter the time limit, time's starting point and step size: ";
    cin >> tmax >> t >> dt;
    cout << "Enter alpha and beta: ";
    cin >> alpha >> beta;

    clock_t tStart = clock();

    ofstream dataFile("multipop.data");

    dn1 = dt * r1 * n1 * (1 - (n1 + alpha * n2) / k1);
    dn2 = dt * r2 * n2 * (1 - (n2 + beta * n1) / k2);


    while (t < tmax || n1 >= k1 || n2 >= k2) {

        t += dt;
        n1 += dn1;
        n2 += dn2;

        dn1 = dt * r1 * n1 * (1 - (n1 + alpha * n2) / k1);
        dn2 = dt * r2 * n2 * (1 - (n2 + beta * n1) / k2);
```

```
        dataFile << t << ' ' << n1 << ' ' << n2 << '\n';

    }

    dataFile.close();

    cerr << "Time elapsed: " << (double)(clock() - tStart)/CLOCKS_PER_SEC << " s" << endl;

}
```

A harmadik feladat megvalósításához írt kód:

```
#include <cmath>
#include <cstdlib>
#include <fstream>
#include <iostream>
using namespace std;


int main() {
    double nR = 500, nF = 500, dnR, dnF, tmax, dt, t, a, b, c, d;

    cout << "Enter the time limit, time's starting point and step size: ";
    cin >> tmax >> t >> dt;
    cout << "Enter the values of a, b, c, d: ";
    cin >> a >> b >> c >> d;

    clock_t tStart = clock();

    ofstream dataFile("LV.data");

    dnR = dt * (a * nR - b * nF * nR);
    dnF = dt * (c * nF * nR - d * nF);


    while (t < tmax) {

        t += dt;
        nR += dnR;
        nF += dnF;

        dnR = dt * (a * nR - b * nF * nR);
        dnF = dt * (c * nF * nR - d * nF);

        dataFile << t << ' ' << nR << ' ' << nF << '\n';

    }

    dataFile.close();

    cerr << "Time elapsed: " << (double)(clock() - tStart)/CLOCKS_PER_SEC << " s" << endl;

}
```

Hivatkozások

[1] Feladatkiírás
`https://stegerjozsef.web.elte.hu/teaching/szamszim/popdin.pdf`

[2] Bolygómozgás - példakód
`https://stegerjozsef.web.elte.hu/teaching/szamszim/`