Populáció dinamika

Szőke Kálmán Benjamin - SZKRADT.ELTE

2012. május 12.

1. Bevezetés

A populációdinamika az élőlények egyedszámának és népességviszonyainak térbeli és időbeli változásának menetét adja meg. Habár a leírás során használt differenciálegyenletek csak közelítését adják meg a valóságnak, mégis érdekes tapasztalatokat és motívumokat kaphatunk meg belőlük.

A populációdinamika tekinthető egy általános keretrendszernek, melyben ragadozók, növényevők és ezek tápláléka szerepel, ám könnyen behelyettesíthetünk szinte ugyanazon egyenletekbe más-más paramétereket (koncentráció, katalizátor), és látszik, hogy a kialakuló dinamika hasonló mintázatokat fog követni.

2. Alapfeltevések és a logisztikus egyenlet

Tekintsünk tehát egy populációt, mely n darab egyedből áll. Ennek változása a t idő függvényében, első ránézésre (ha más befolyásoló tényező nincs) a populáció létszámával arányos lesz. Legyen a az adott populációra jellemző szaporodási ráta, vagyis Δt idő alatt ennyi utód jön világra.

$$n_t + \Delta t = n_t + an_t$$

Ha Δt időt elég rövid tartományra nézzük, akkor határesetben a fenti egyenlet a

$$\frac{dn}{dt} = an$$

kifejezésbe megy át. A megoldás pedig a valóságban is tapasztalt exponenciális növekedést fogja megadni.

$$n = \exp at$$

Persze a valóságban egy populáció növekedése sokkal több tényezőtől függ, vizsgáljuk meg először, hogy hogyan változik az egyedszám a d halálozási ráta bevezetésével.

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

Ebben az esetben megmarad az exponenciális görbe, azonban értékkészlete alapján ez csökkenő vagy növekedő lesz, ha konstans létszámú populáció kapunk, akkor az nem lesz stabil. Második közelítésként persze bevezethetjük az élelemforrás korlátosságát, ennek hatását a populáció egyedszámának változására egy szorzótényezővel szemléltethetjük.

$$\frac{dn}{dt} = rnF\left(n\right)$$

Ha feltevésünk szerint a rendelkezésre álló élelemkészlet elegendő egy maximálisan k (kapacitás) létszámú populáció ellátásához, akkor F(k)=0, tehát F() nem enged további szaporodást (kis populáció esetében F() nem módosítja az eredeti összefüggést F(0)=1). Ezt a feltételt az F(n)=1-n/k egyenlet elégíti ki. Behelyettesítve tehát az eredeti differenciálegyenlet az alábbi kifejezésre módosul.

$$\frac{dn}{dt} = rn\left(1 - \frac{n}{k}\right)$$

Ezt nevezzük a populációdinamika logisztikus egyenletének, amely megfelelő skálázás mellett n/k=x a következő alakra hozható.

$$\frac{dx}{dt} = rx\left(1 - x\right)$$

Ennek megoldása r-től és x0 kiindulási értékétől függően csökkenő vagy növekvő szigmoid jellegű görbe.

$$x\left(t\right) = \frac{1}{1 + \left(\frac{1}{x_0} - 1\right)\exp -rt}$$

A logisztikus egyenlet tehát egy nemlineáris differenciálegyenlet, és ott vannak úgynevezett fixpontjai, ahol a dx/dt=0. Jelen esetünkben ez x=0 és x=1 helyen van. Ezek a pontok lehetnek stabilak és instabilak is, hiszen ha a fixpontból kitérítjük visszatérhet oda vagy exponenciálisan eltávolodhat a megoldás. Látható tehát, hogy ezek vizsgálata perturbációkkal a legegyszerűbb. Legyen a differenciálegyenlet a

$$\frac{dx}{dt} = f\left(x\right)$$

alakú.

Innen kis perturbációval kitérítve kapjuk a

$$x(t) = x_* + \epsilon(t)$$

alakú lineáris közelítést. Ezt Taylor-sorba fejtve, majd a magasabb rendű tagok elhagyásával kapjuk a

$$\epsilon(t) = \epsilon(0) \exp f'(x_*)\, t$$

alakú megoldást. Ez akkor lesz stabil, ha $t \longrightarrow 0$ esetben $\epsilon(t) \longrightarrow 0$.

3. Két faj versengése

Ha egy adott területen két faj küzd egyazon táplálékért, akkor ezek kölcsönhatásba lépnek egymással és a gyorsabban szaporodó, vagy valamely más tulajdonságát tekintve előnyben lévő faj akár ki is szoríthatja a másikat a területről teljesen. Legyen a két faj differenciálegyenlete:

$$\frac{dn_1}{dt} = r_1 n_1 \left(1 - \frac{n_1}{k_1}\right)$$
$$\frac{dn_2}{dt} = r_2 n_2 \left(1 - \frac{n_2}{k_2}\right)$$

A versengés szerint, ha az egyik faj szaporodik, akkor mindkét faj számára kevesebb eleség jut, azaz az egyenlet jobb oldala negatív részében jelentkezik a csatolás:

$$\frac{dn_1}{dt} = r_1 n_1 \left(1 - \frac{n_1 + \alpha n_2}{k_1}\right)$$
$$\frac{dn_2}{dt} = r_2 n_2 \left(1 - \frac{n_2 + \beta n_1}{k_2}\right)$$

Itt α és β dimenzió nélküli paraméterek, kifejezi az egyes fajok élelemfogyasztásának arányát. Az egyenletekben szereplő paraméterek más-más értékeinél vizsgáljuk a kialakuló rendszereket.

4. Eredmények

4.1. Euler-módszer, Runge-Kutta módszer, Analitikus megoldás

Első feladat a numerikus és analitikus megoldások különbségének ábrázolása volt, vagyis hogy az eljárások mennyire különböznek egymástól. Az Euler-módszert és az adaptív Runge-Kutta módszert hasonlítottam össze az analitikus megoldással.

4.1.1. Euler-módszer

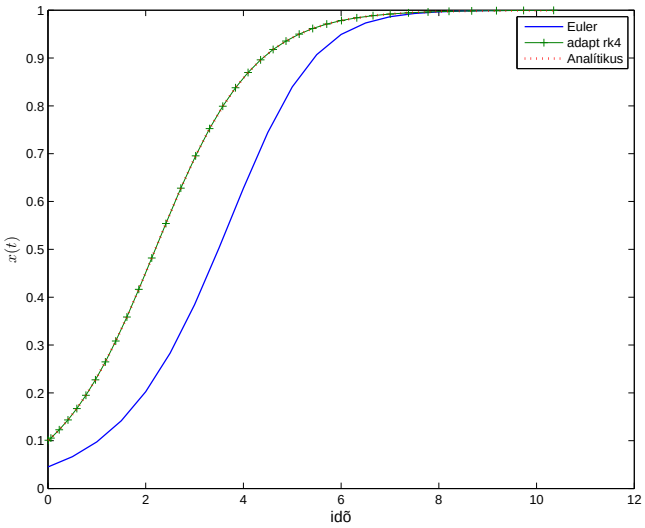
```
#include <cmath>
#include <fstream>
#include <iostream>
#include <string>
using namespace std;
int main(){
cout << "Euler modszer populacio dinamika" << "Add meg r-t es y(0)-t: ";
double r, y, tMax, dt, t;
string FileName;
cin >> r >> y;
cout << "Add meg time t_max-t: ";
cin >> tMax;
cout << "Add meg a kimeneti fájl nevet: ";
cin >> FileName;
cout << " Add meg t_0-t: ";
cin >> t;
cout << " Add meg dt-t: ";
cin >> dt;
  ofstream dataFile(FileName.c_str());
y=dt*r*y*(1-y);
dataFile << t << '\t' << y << '\n';
while (t < tMax){
t+=dt;
y+=dt*r*y*(1-y);
dataFile << t << '\t' << y << '\n';
}
dataFile.close();
}
```

4.1.2. Runge-Kutta módszer

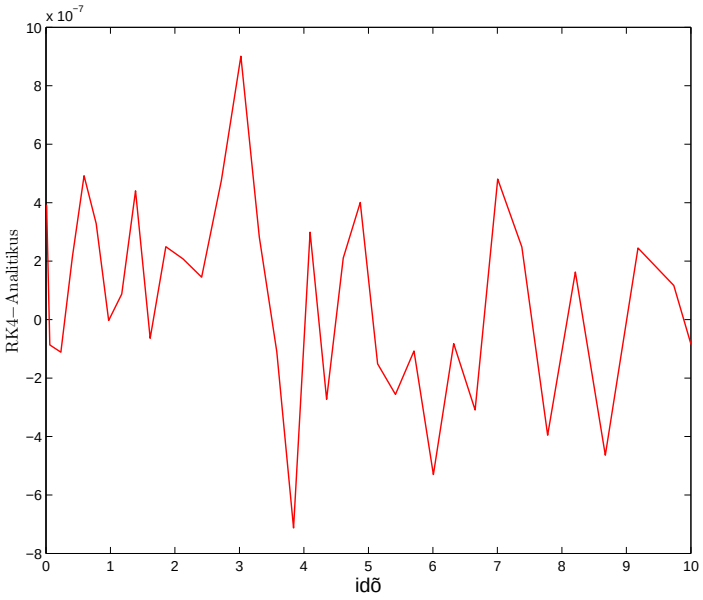
A Runge-Kutta módszerhez az inga szimulációja során is használt vector.hpp-t, és odeint.hpp-t használtam.

```
#include <cmath>
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
#include "vector.hpp"
#include "odeint.hpp"
```

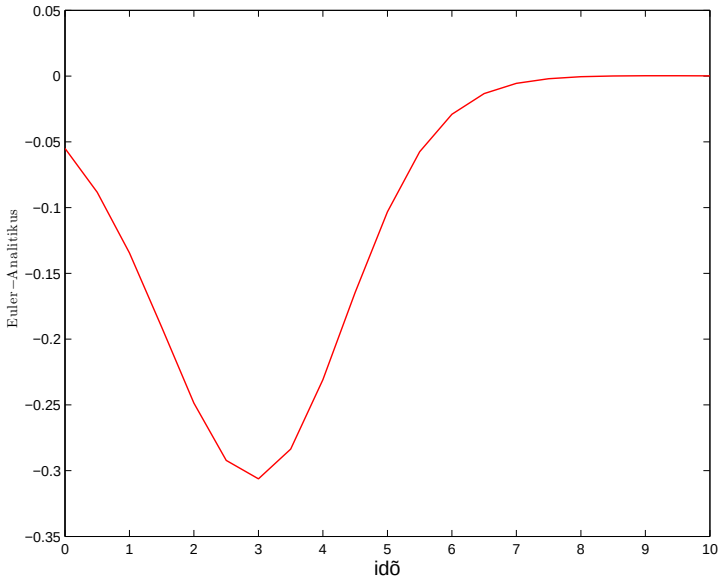
```
using namespace cpl;
const double pi = 4 * atan(1.0);
Vector f(const Vector& x) {
  double t = x[0], y = x[1], r=1;
  Vector f(2);
  f[0] = 1;
  f[1] = r*y*(1-y);
  return f;
}
int main() {
string FileName;
  cout << "Add meg r-t es y(0)-t: ";
  double y, r, tMax;
  cin >> y >> r;
  cout << "Add meg t_max-t: ";
  cin >> tMax;
cout << "Add meg a kimeneti fájl nevet: ";
cin >> FileName;
  double dt = 0.01, accuracy = 1e-6, t=0;
  ofstream dataFile(FileName.c_str());
  Vector x(2);
  x[0] = t;
  x[1] = y;
  while (t < tMax) {
      adaptiveRK4Step(x, dt, accuracy, f);
      t = x[0], y = x[1];
      dataFile << t << '\t' << y << '\n';
  }
  dataFile.close();
}
```



1. ábra. RK4, Euler, Analitikus



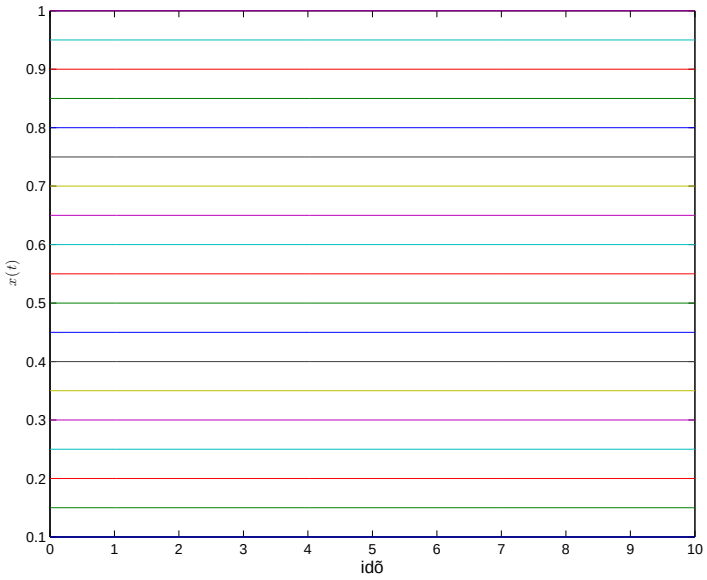
2. ábra. RK4 − Analitikus



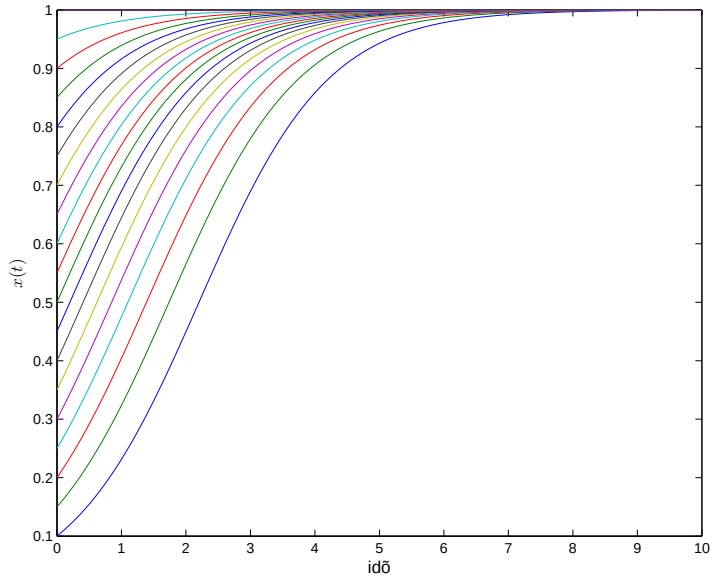
3. ábra. Euler − Analitikus

4.1.3. Csatolt logisztikus modell

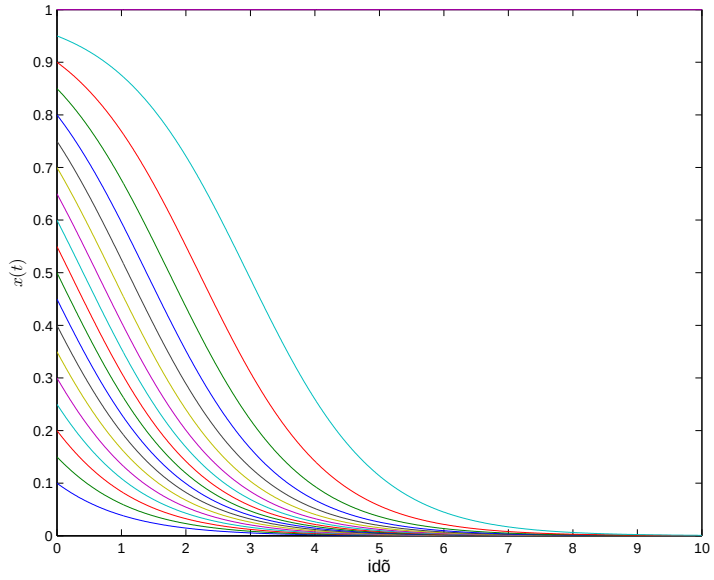
Az Euler-módszerrel az eredmények jól vissza adták a jegyeztben megtalálható logisztikus egyenlet megoldásait, különböző r értékekre.



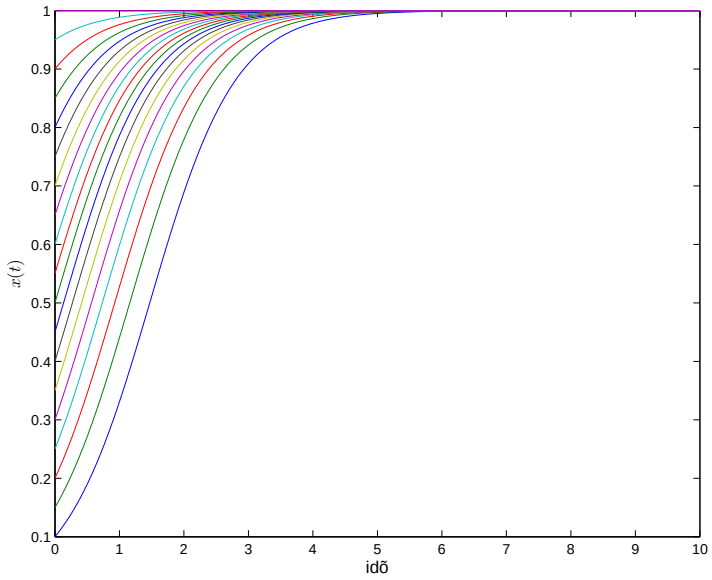
4. ábra. $r = 0$



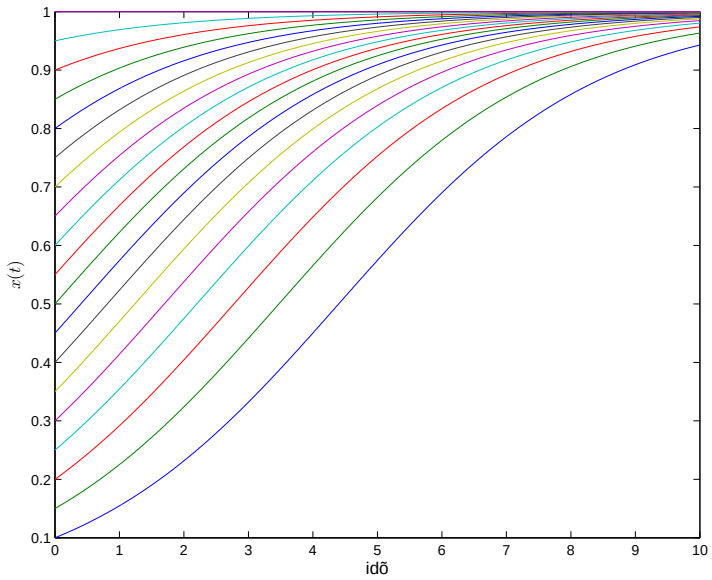
5. ábra. $r = 1$



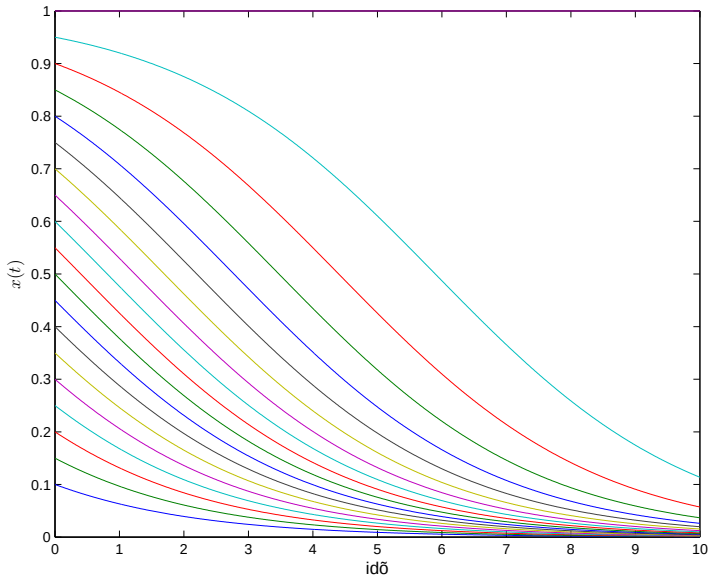
6. ábra. $r = -1$



7. ábra. $r = 1.5$



8. ábra. $r = 1$



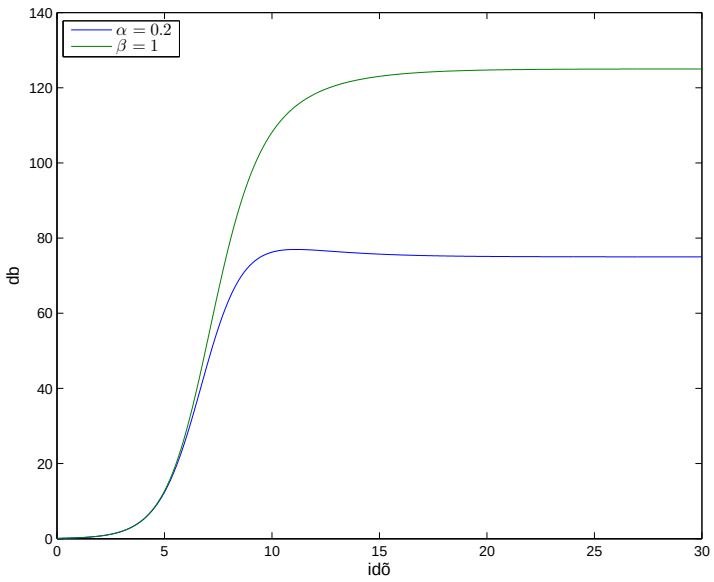
9. ábra. $r = -0.5$

4.2. Fajok erőforrásért való versengése

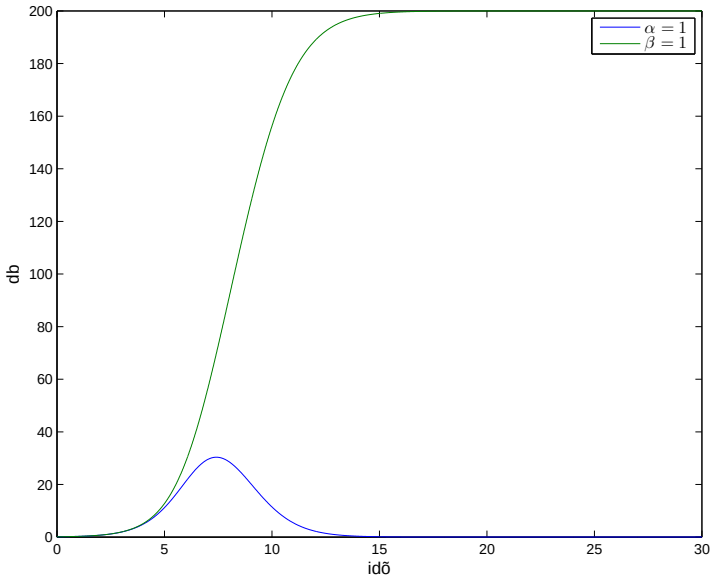
A szimuláció során igazolódik a kompetitív kizárás törvénye, vagyis $\alpha = \beta = 1$ esetén nem létezhet stabilan két faj. A két faj együttélése csak $\alpha k_2 < k_1$ és $\beta k_1 < k_2$ esetén valósúl meg.

```
#include <cmath>
#include <fstream>
#include <iostream>
#include <string>
using namespace std;
int main(){
cout << "Euler modszer populacio dinamika" << "Add meg r-t es y(0)-t: ";
double r, y, tMax, dt, t, r1, r2, y1, y2;
string FileName;
cin >> r >> y;
r1=r;
r2=r;
y1=y;
y2=y;
cout << "Add meg alfa es beta erteket: ";
double alfa, beta;
cin >> alfa >> beta;
cout << "Add meg k1 es k2 erteket: ";
double k1, k2;
cin >> k1 >> k2;
cout << "Add meg time t_max-t: ";
cin >> tMax;
```

```
cout << "Add meg a kimeneti fájl nevet: ";
cin >> FileName;
cout << " Add meg t_0-t: ";
cin >> t;
cout << " Add meg dt-t: ";
cin >> dt;
  ofstream dataFile(FileName.c_str());
y1=y1+r1*y1*dt*(1-(y1+alfa*y2)/k1);
y2=y2+r2*y2*dt*(1-(y2+beta*y1)/k2);
dataFile << t << '\t' << y1 << y2 << '\n';
while (t < tMax){
t+=dt;
y1+=y1+r1*y1*dt*(1-(y1+alfa*y2)/k1);
    y2+=y2+r2*y2*dt*(1-(y2+beta*y1)/k2);
dataFile << t << '\t' << y1 << y2 << '\n';
}
dataFile.close();
}
```



10. ábra. Stabil faj

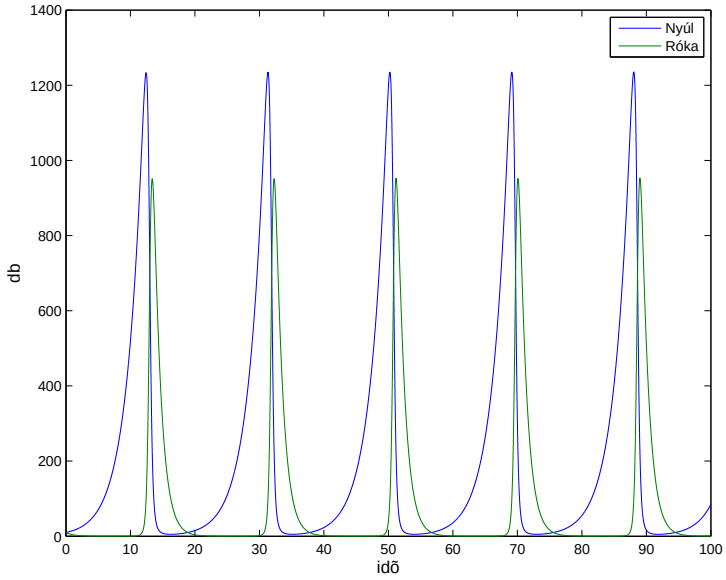


11. ábra. Instabil faj

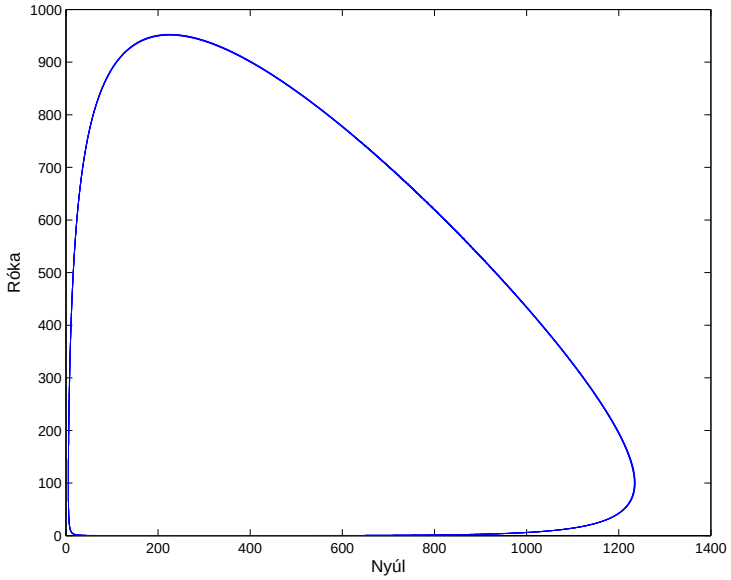
4.3. Lotka-Volterra modell

```
#include <cmath>
#include <fstream>
#include <iostream>
#include <string>
using namespace std;
int main(){
cout << "Euler modszer populacio dinamika" << "Add meg R-t es F-t: ";
double R, F, tMax, dt, t;
string FileName;
cin >> R >> F;
cout << "Add meg a es b erteket: ";
double a, b;
cin >> alfa >> beta;
cout << "Add meg c es d erteket: ";
double c, d;
cin >> c >> d;
cout << "Add meg time t_max-t: ";
cin >> tMax;
cout << "Add meg a kimeneti fájl nevet: ";
cin >> FileName;
cout << " Add meg t_0-t: ";
cin >> t;
cout << " Add meg dt-t: ";
cin >> dt;
  ofstream dataFile(FileName.c_str());
R=R+a*R*dt-b*R*F*dt;
  F=F-d*F*dt+c*F*R*dt;
dataFile << t << '\t' << R << F << '\n';
```

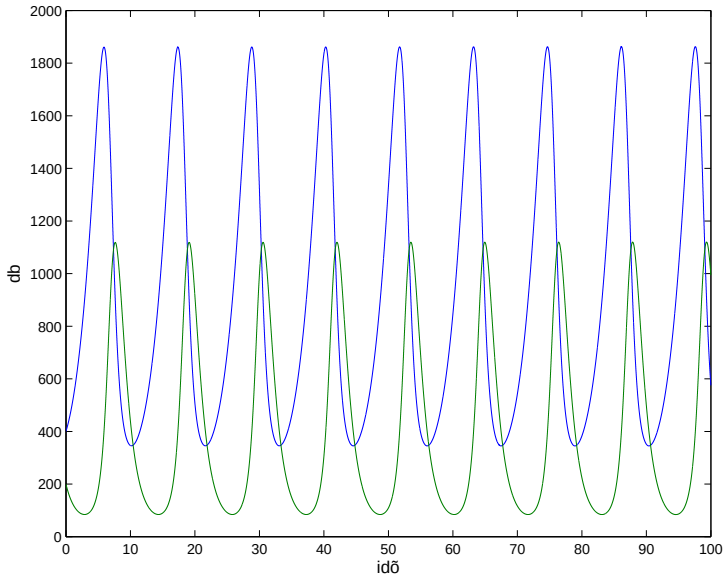
```
while (t < tMax){
t+=dt;
R+=R+a*R*dt-b*R*F*dt;
    F+=F-d*F*dt+c*F*R*dt;
dataFile << t << '\t' << R << F << '\n';
}
dataFile.close();
}
```



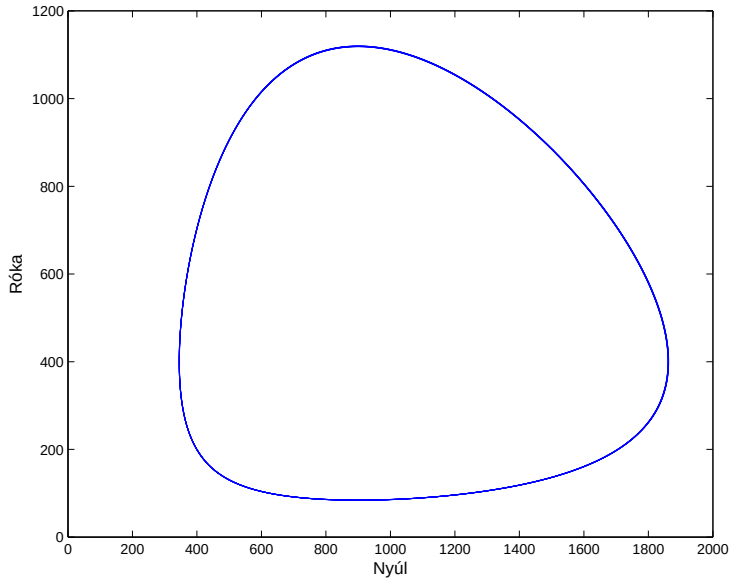
12. ábra. $b = 0.004$, $c = 0.004$



13. ábra. $b = 0.004$, $c = 0.004$



14. ábra. $b = 0.001$, $c = 0.001$



15. ábra. $b = 0.001$, $c = 0.001$

Tartalomjegyzék

1. Bevezetés **1**

2. Alapfeltevések és a logisztikus egyenlet **1**

3. Két faj versengése **3**

4. Eredmények **4**

4.1. Euler-módszer, Runge-Kutta módszer, Analitikus megoldás . . 4

4.1.1. Euler-módszer 4

4.1.2. Runge-Kutta módszer 4

4.1.3. Csatolt logisztikus modell 7

4.2. Fajok erőforrásért való versengése 10

4.3. Lotka-Volterra modell 12

Hivatkozások

[1] Jegyzet
http://complex.elte.hu/∼csabai/szamszim/simLec6.pdf