SZÁMÍTÓGÉPES SZIMULÁCIÓK

Populációdinamika

jegyzőkönyv

Jegyzőkönyvet készítette:
Koroknai Botond (AT5M0G)

Jegyzőkönyv leadásának időpontja:
2024.05.07

1. Probléma rövid ismertetése

Egy populáció létszámát (n) szeretnénk vizsgálni az idő (t) függvényében. Ha nincs semmilyen tényező, akkor feltételezhetjük, hogy a szaporodás a populáció létszámával arányos. Vezessük be a szaporodási rátát (a), amivel megadhatjuk, hogy Δt idő alatt mennyi utód jön világra:

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

Ha Δt kicsi, akkor határesetben a fenti egyenlet:

$$\frac{dn}{dt} = an \tag{2}$$

Ezen egyenlet megoldása a jól ismert exponenciális növekedés lesz:

$$n = e^{at} \tag{3}$$

Mivel természetesen nem csak növekedésről beszélhetünk, a modell realisztikusabb lesz ha bevezetnük egy d halálozási rátát:

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

Így a megoldás $r = a-d$ előjelétől függően exponenciálisan növekvő vagy csökkenő lesz. Ezt követően vegyük figyelembe, hogy az erőforrások korlátosak, ennek hatását az egyedszám változására egy szorzótényezővel szemléltethetjük:

$$\frac{dn}{dt} = rnF(n) \tag{5}$$

Ha most feltesszük, hogy az erőforrások egy k létszámú populációt tudnak fenntartani, akkor a következő feltételt írhatjuk fel:

$$F(n)1 - \frac{n}{k} \tag{6}$$

A differenciál egyenlet így:

$$\frac{dn}{dt} = rn\left(1 - \frac{n}{k}\right) \tag{7}$$

A loisztikus egyenlet felírásához skálázzuk át $x = n/k$-val a kifejezést, ekkor:

$$\frac{dx}{dt} = rx(1 - x) \tag{8}$$

A megoldás így r-től és a kezdeti x_0 tól függően növekedő vagy csökkenő szigmoid jellengű görbe:

$$x(t) = \frac{1}{1 + \left(\frac{1}{x_0} - 1\right) e^{-rt}} \tag{9}$$

Az egyenlet fixpontjai ott lesznek ahol a $\frac{dx}{dt} = 0$. Ezek lehetnek stabilak vagy instabilak. A stabilitást a legegyszerűbben perturbációkkal tudjuk vizsgálni. Legyen a diffegyenlet

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

alakú, és egy fix pontja x*, azaz ahol f(x*) = 0. Kis perturbációval kimozdítva a rendszert, a megoldást lineáris közelítésben kereshető:

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

Visszaírva az eredeti egyenletbe és Taylor sorba fejtve:

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

A magasabb rendű deriváltakat hagyjuk el:

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

amit megold:

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

2. Feladatok

2.1. 1. Feladat

Az első feladat annak a vizsgálata volt, hogy mennyire pontosan adja vissza az Euler-módszer, valamint az adaptív Runge-Kutta az analitikusan kiszámolható (9. képlet) függvény értékeit. Ehhez először implementálni kellett a két numerikus megoldás függvényét.

Logisztikus egyenlet: A nulladik lépésként definiáltam a logisztikus egyenletet (8. képlet), melyet később mindkét megvalósítás során fel fogok használni.

1. kód. Logisztikus egyenlet

```
double logistic_eq(double x, double r)
{
    return r * x * (1 - x);
}
```

Így az **Euler módszer** a következő alakot fogja ölteni:

2. kód. Euler módszer

```
void euler(double x0, double r, double dt, int steps, const char *filename)
{
    double x = x0;
    ofstream file(filename);

    for (int i = 0; i < steps; ++i)
    {
        x += dt * logistic_eq(x, r);
        file << i * dt << " " << x << endl;
    }

    file.close();
}
```

Azaz meghatározzuk a változás sebességét a *logistic_eq* függvénnyel, majd megszorozzuk ezt az időlépéssel (dt), és hozzáadjuk az aktuális populáció méretéhez (x).

Az **adaptív Runge-Kutta** megvalósításához egy közönséges negyrendű Runge-Kutta algortimust valósítottam meg:

3. kód. Adaptív Runge-Kutta

```
void runge_kutta(double x0, double r, double dt, int steps, const char *filename)
{
    double x = x0;
    ofstream file(filename);

    for (int i = 0; i < steps; ++i)
    {
        double k1 = dt * logistic_eq(x, r);
        double k2 = dt * logistic_eq(x + k1 / 2, r);
        double k3 = dt * logistic_eq(x + k2 / 2, r);
        double k4 = dt * logistic_eq(x + k3, r);
        x += (k1 + 2 * k2 + 2 * k3 + k4) / 6;
        file << i * dt << " " << x << endl;
    }

    file.close();
}
```

Az **analitikus megoldást** pythonban valósítottam meg, egyszerűen a 9. képletet meghívva egy függvényként:

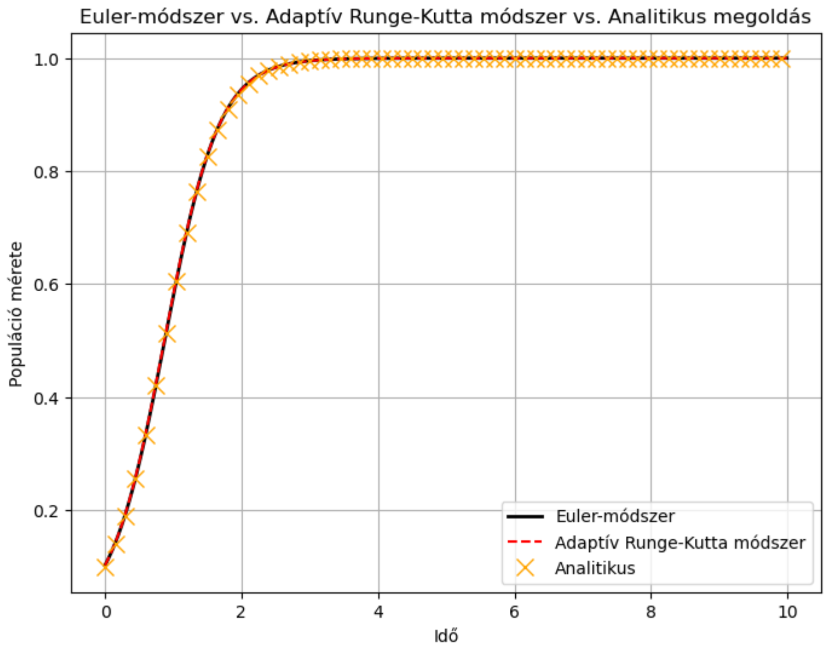
```python
def logistic_analytical(t, x0, r):
    return 1 / (1 + (1 / x0 - 1) * np.exp(-r * t))
```

4. kód. Analitikus megoldás

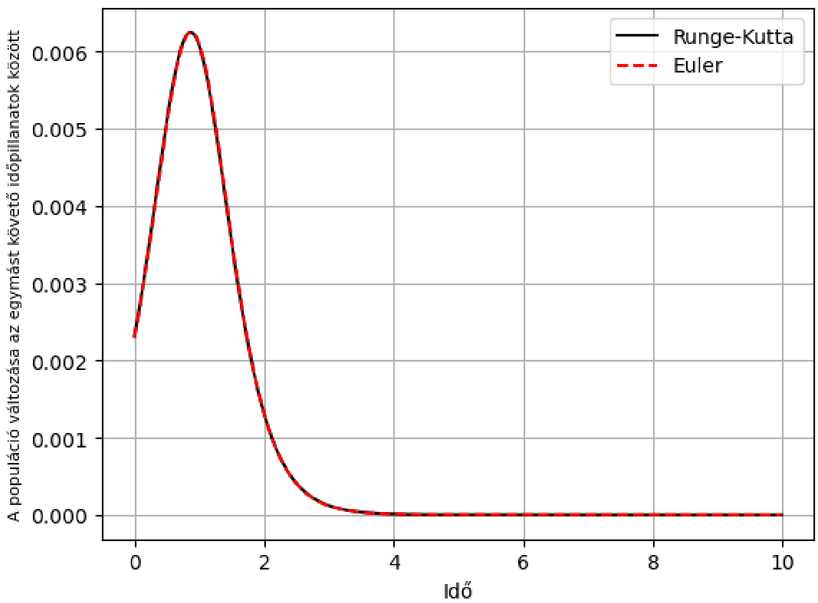
A szimuláció megvalósításához a következő kezdeti paramétereket választottam:

- x_0 - kezdeti populáció = 0.1
- r - szaporodási ráta = 2.5
- dt - időlépés = 0.01
- steps - lépés = 1000

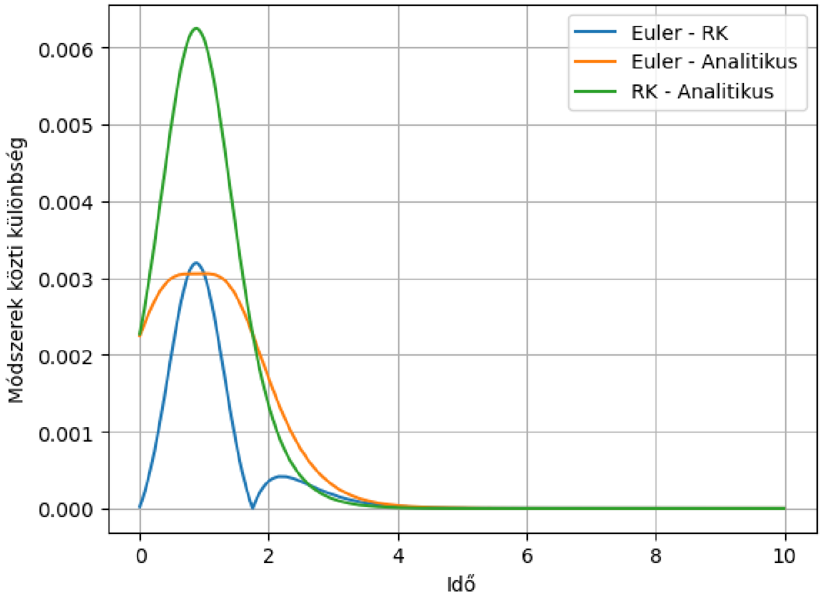
A szimuláció futtatását követően a következő eredményt kaptam:



Mivel a feladat az volt, hogy a különböző módszerek viselkedését vizsgáljuk a fix pontokhoz közel, illeve azoktól távol, így következőnek megkerestem a fixpontokat. Mivel definíció szerint azok a fix pontok ahol a fügvény deriváltja nulla, ezért végig iteráltam a szimuláció alatt keletkező adatosorokon és minden $i+1$ elemből kivontam a i. elemeket ezzel vizsgálva, hogy változik-e a függvény.



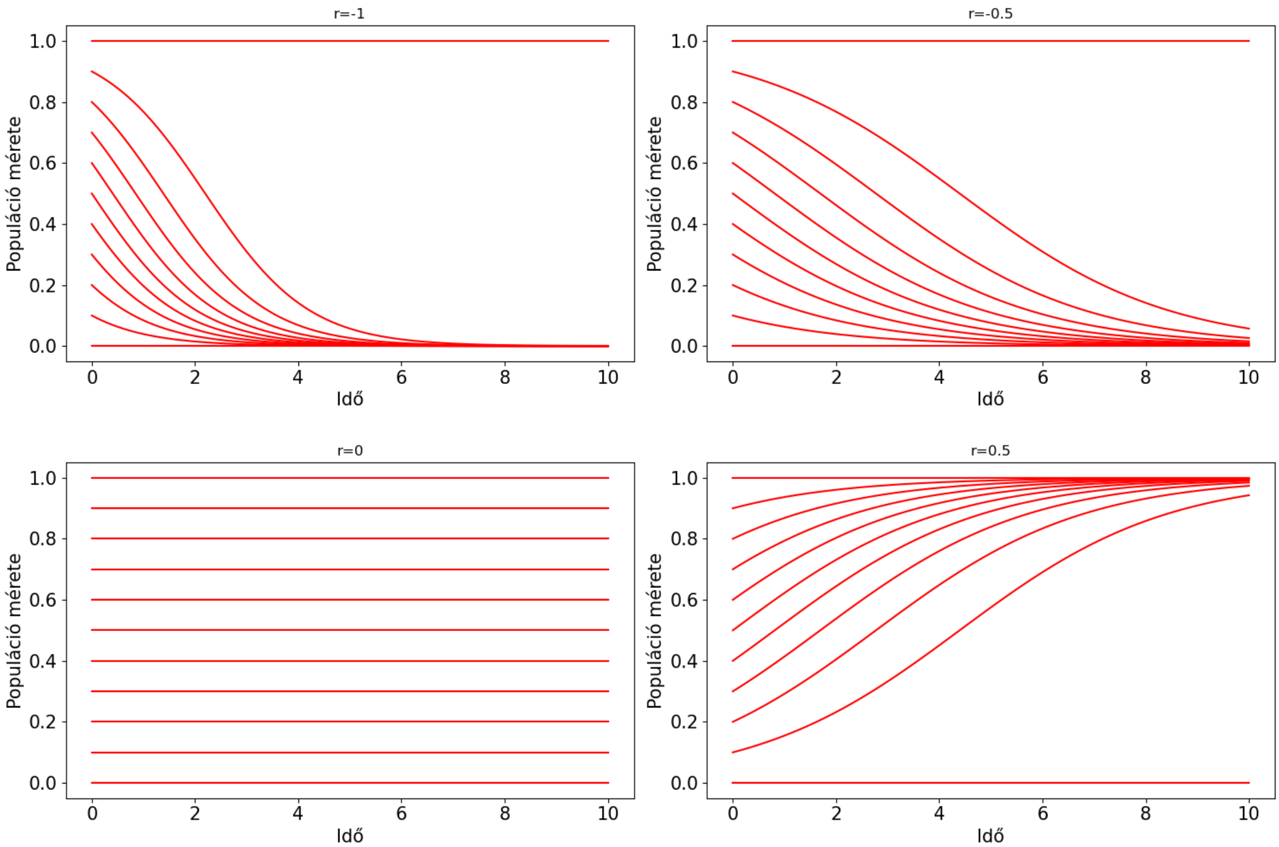
Ezt követően ábrázoltam a a módszerek közti abszolút különbségeket:

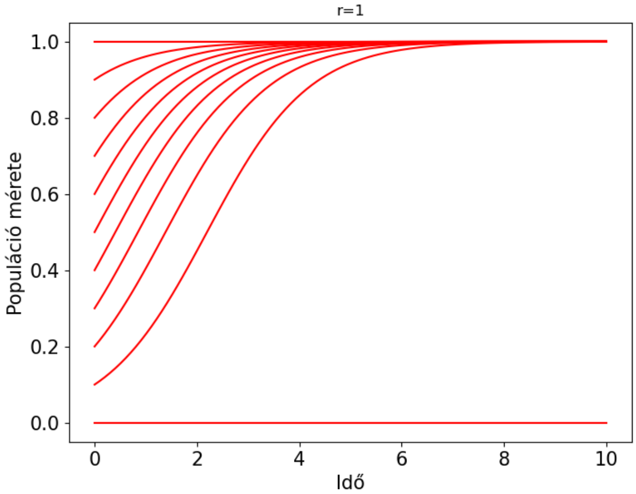


Ami alapján azt a következtetést vonhatjuk le, hogy a fixpontoktól távol, ahol változik a függvény, ott a numerikus megoldások eltérnek az analitikustól, viszont a fixpontok közelében, mind a három módszer stabilizálódik.

2.1.1. Fólia ábrák

A diasor alapján a fólia ábrákat 5 különböző (-1,-0.5,0,0.5,1) szaporodási ráta melett készítettem el:





2.2. 2. feladat

2.2.1. Probléma ismertetése

Ha egy élőhelyen két faj küzd a táplálékért, véges erőforrások melett, akkor az egymáshoz viszonyított szaporodási rátájuk és a környezet eltartóképessége függvényében a "rátermetebb" faj akár teljesen elfoglalhatja a nichet. Ez matematikailag a következőképpen fogalmazható meg:

$$\frac{dn_1}{dt} = r_1 n_1 \left(1 - \frac{n_1 + \alpha n_2}{k_1}\right) \tag{10a}$$

$$\frac{dn_2}{dt} = r_2 n_2 \left(1 - \frac{n_2 + \beta n_1}{k_2}\right) \tag{10b}$$

Ahol α és β paraméterek azt felyezik ki, hogy milyen arányban fogyasztja az egyik faj a másik erőforrásait.

2.2.2. Szimuláció

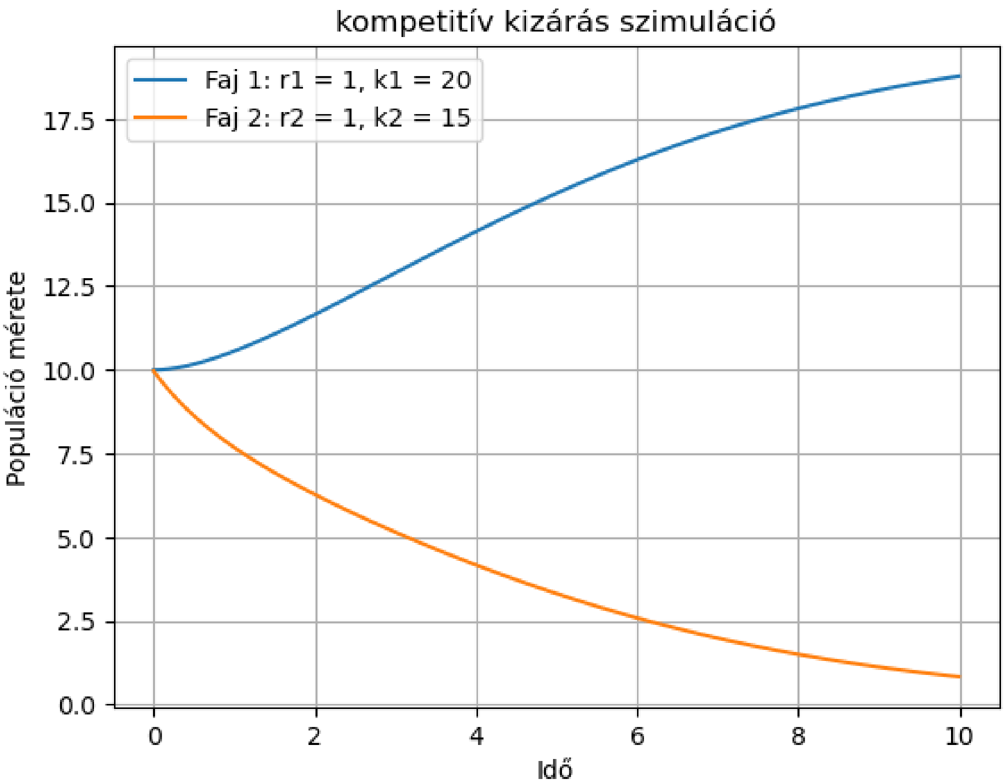
Az első feladat az volt, hogy demonstráljuk a kompetetív kizárás törvényét, azaz két faj nem létezhet stabilan együtt, a nagyobb k értékű kiszorítja a másikat.

A szimuláció megvalósításához a 10a és 10b képletek alapján egy Euler algortimust implementáltam.

5. kód. Versengés az erőforrásért

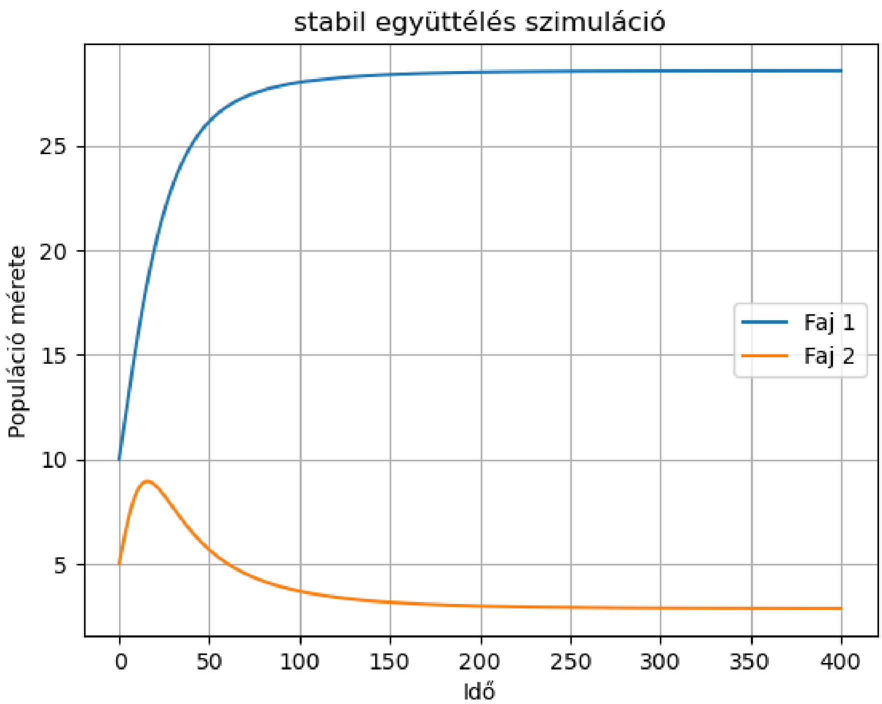
```
void competitive_exclusion(double n1_0, double n2_0, double r1, double r2, double k1, doub
{
    double n1 = n1_0;
    double n2 = n2_0;
    ofstream file(filename);

    for (int i = 0; i < steps; ++i)
    {
        double dn1 = r1 * n1 * (1 - (n1 + n2) / k1);
        double dn2 = r2 * n2 * (1 - (n1 + n2) / k2);
        n1 += dt * dn1;
        n2 += dt * dn2;
        file << i * dt << " " << n1 << " " << n2 << endl;
    }

    file.close();
}
```

Az elméletnek megfelelően a nagyobb k - eltartási paraméterrel rendelkező faj kezdett el szaporodni, míg a másik populációja csökkent.



A következő feltevés amit meg kellett vizsgálni az volt, hogy két faj együttélése csak akkor lehet stabil ha $\alpha k_2 < k_1$ és $\beta k_1 < k_2$. Ennek megfelelően a paramétereket a következőnek választottam meg:

- $\alpha = 0.5$
- $\beta = 0.6$
- $k_1 = 30$
- $k_2 = 20$



Mint látjuk kellő idő elteltével valóban stabilizálódik a két faj populációjának mérete.

2.3. 3. feladat

2.3.1. Probléma ismertetése

A fajok kölcsönhatása nem csak a közös erőforrásokért való küzdelemben nyilvánulhat meg. Ilyen például a nyulak és rókák esete.

Az ilyen típusú eseteket a Lotka-Volterra modellel írhatjuk le:

$$\frac{dn_R}{dt} = an_R - bn_F n_R \tag{11a}$$

$$\frac{dn_F}{dt} = cn_R n_F - dn_F \tag{11b}$$

Ahol ahol n_R a nyulak és n_F a rókák száma, a a nyulak szaporodási, és bn_F a halálozási rátájuk, míg d és cn_R a rókák pusztulási , és szaporodási rátája.

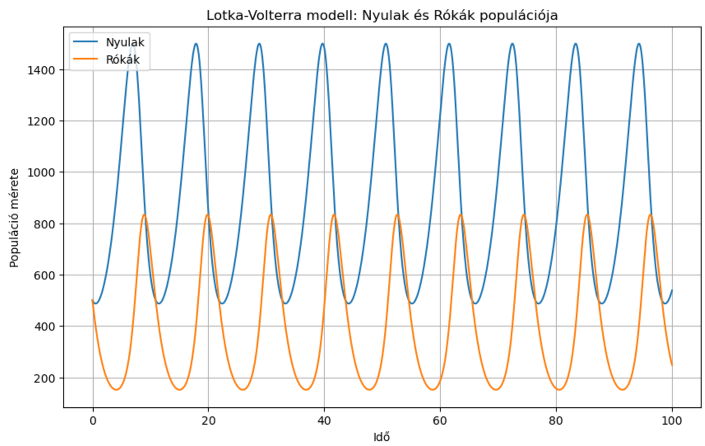
Ezen felül még egy szimulációt kell, hogy elvégezzünk, ahol a modellt még realisztikusabbá tehetjük azzal, ha korlátozzuk a nyulak táplálék forrását, valamint a rókák nyúlfogyasztási képességét:

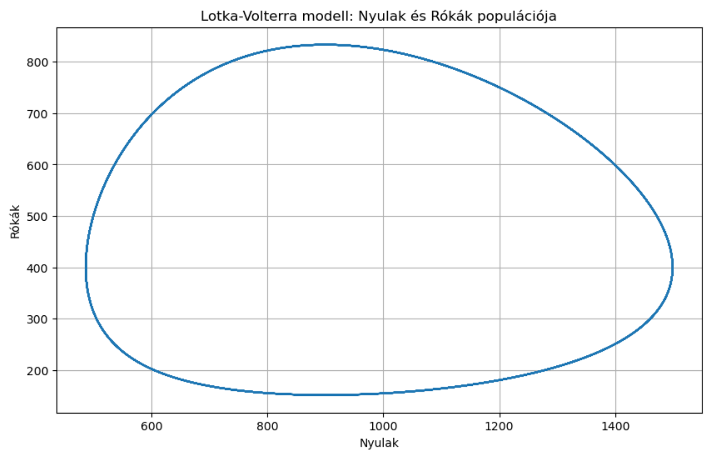
$$a \to a\left(1 - \frac{n_R}{k}\right) \tag{12}$$

$$n_R n_F \to \frac{n_R n_F}{1 + n_r S} \tag{13}$$

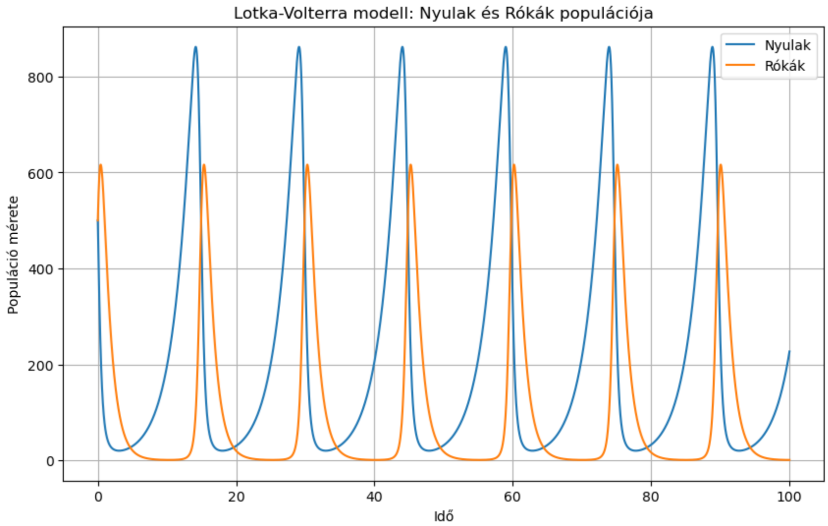
2.3.2. Szimuláció

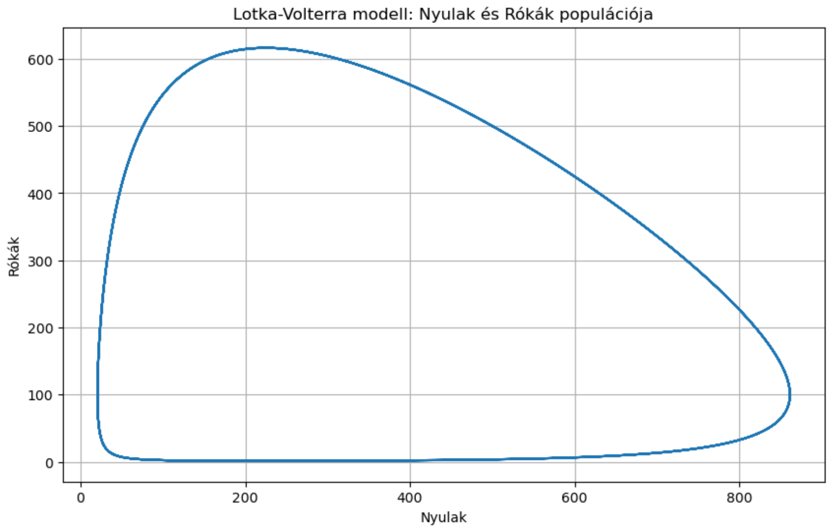
A szimuláció megvalósításához először Euler módszert alkalmaztam, majd mikor kiértékelés során instabil eredményt kaptam, megismételtem a szimulációt az adaptív Runge-Kutta segítségével. A szimulációkhoz először az $a = 0.4$, $b, c = 0.001$ és $d = 0.9$ paramétereket használtam fel:



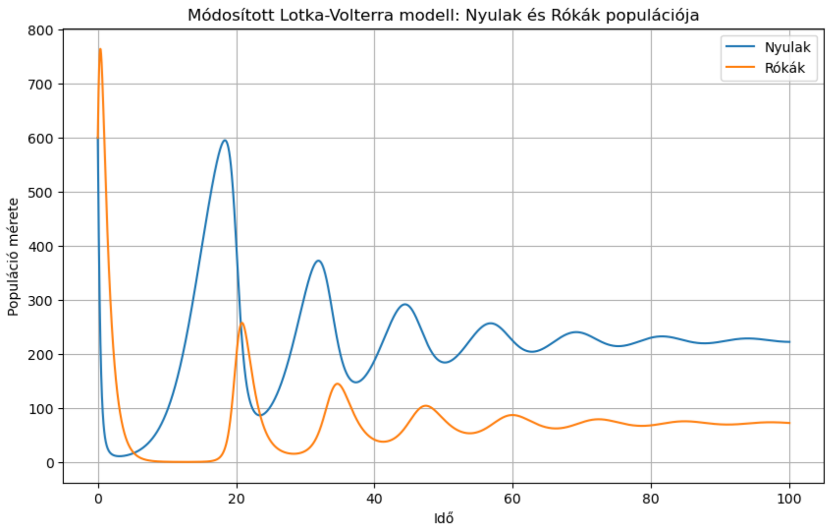


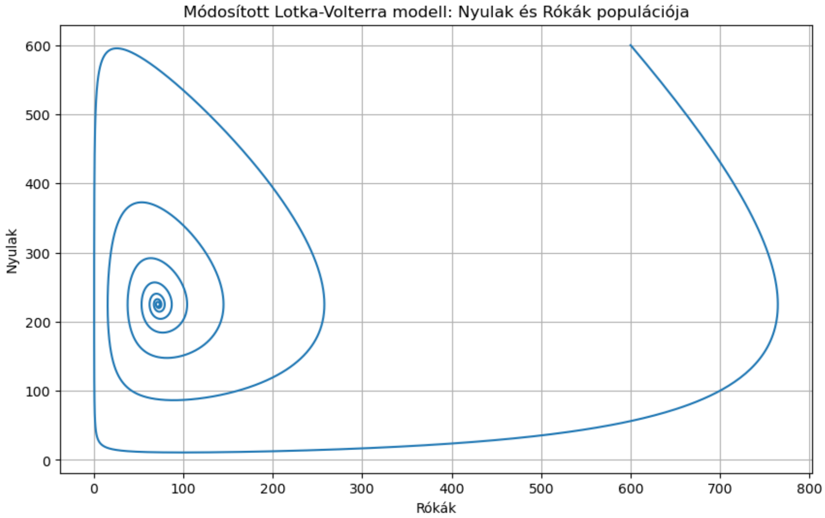
Ezt követően megismételtem ugyan ezt a szimulációt a módosított paraméterekkel: $a = 0.4$, $b, c = 0.004$ és $d = 0.9$





Ez után létrehoztam a módosított Lotka-Volterra modellt is, ami figyelembe veszi a 12-es és 13-as képletben bemutatott korlátozásokat, A paraméterek amiket haszálntam: $a = 0.4$, $b, c = 0.004$, $d = 0.9$, valamint $K = 800$ értékeket állítottam be.





3. Megjegyzés

Az egyes feladatokhoz tartozó kódok a feladat számának megfelelő .cpp fájlokban találhatóak.

Az elméleti hátteret a Poplációdinamika c. diasor alapján készítettem.