Lotka-Volterra

Obyčejné diferenciální rovnice: Lotka-Volterra

Tomáš Kalvoda, 2014

Formulace problému

Lotka-Volterrův problém (*predator-prey* problém) je soustava dvou nelineárních obyčejných diferenciálních rovnic tvaru

$$\begin{aligned} x' &= \alpha x - \beta xy, \\ y' &= \delta xy - \gamma y, \end{aligned}$$

kde funkce y a x představují množství predátorů (např. lišek) a obětí (např. králíků). Nezávisle proměnnou je čas t v němž se systém vyvýjí. V zadání úlohy je dále nutné specifikovat počáteční podmínky $x(0)$ a $y(0)$, tedy počáteční počty lišek a králíků. Koeficienty α, β, γ a δ jsou kladné a členy vyskytující se na pravé straně rovnic mají následující význam:

- αx: rychlost množení králíků,
- $-\beta xy$: úbytek králíků v důsledku jejich lovu liškami (interakce),
- δxy: přírustek lišek závisející na množství potravy, tedy králíků (interakce),
- $-\gamma y$: rychlost vymýrání lišek.

Další zajímavé informace o historii a významu této soustavy obyčejných diferenciální rovnic lze nalézt na wiki. Zde aspoň poznamenejme, že se jedná o dobrou idealizaci. S podobnými systémy, komplikovanějšími systémy, se lze setkat v matematických modelech popisujících biologické, chemické, či ekonomické jevy.

Nalézt explicitní analytické řešení problému uvedeného výše nelze. Je nutné problém řešit přibližně numericky, to provedeme o několik odstavců níže. Na tomto místě ale poznamenejme, že z rovnic můžeme dostat jednu velmi důležitou kvalitativní informaci o jejich řešení. Existuje časově neměnné řešení? Tj. existuje pro zadané konstanty modelu (α, β, γ a δ) konstantní řešení, $x(t) = x_*$, $y(t) = y_*$? Takovéto hodnoty by musel nulovat pravé strany diferenciálních rovnic (časová změna v x a y by pak byla nulová). Muselo by tedy platit

$$\begin{aligned} \alpha x_* - \beta x_* y_* &= 0, \\ \delta x_* y_* - \gamma y_* &= 0. \end{aligned}$$

Řešením je zřejmě

$$y_* = \frac{\alpha}{\beta}\,, \quad x_* = \frac{\gamma}{\delta}\,.$$

Pro tyto počty králíků (x_*) a lišek (y_*) se populace nebudou měnit v čase.

Implementace

Definujme pravou stranu. z zde bude označovat vektor, který má složky (x, y). Dále pravé straně předáváme čtyři parametry modelu v listu **coeffs**.

```
def rhs(z, coeffs):
    """ Pravá strana Lotka-Volterra problému. """
    return vector(
         [coeffs[0]*z[0] - coeffs[1]*z[0]*z[1],
         coeffs[2]*z[0]*z[1] - coeffs[3]*z[1]]
         )
```

Abych výklad drželi co nejjednodušší, vyřešíme problém numericky pomocí Runge-Kutta metody 4. řádu (další informace viz např. zde). Tato implementace samozřejmě není nejefektivnější ani nejrychleší. Ukazuje ale, že rovnici můžeme numericky řešit i snadno vlastními silami a nemusíme se odvolávat na nástroje dostupné v Sage (nebo v *Mathematica*).

```
def nsolve(z0, coeffs, tstep, tmax, recstep):
    """
    Jednoduchá implementace Runge-Kutta metody 4. řádu pro
Lotka-Volterrův problém.

    z0: počáteční hodnoty
    coeffs: parametry pravé strany
    tstep: pevný časový krok numerické integrace
    tmax: délka časového intervalu na kterém řešení počítáme
    recstep: časové kroky v kterých zaznamenáváme hodnoty
numerických aproximací

    výstup: list elementů [čas, [hodnota x, hodnota y]]
    """
    t = 0.0; trec = 0.0;
    z = vector(z0)
    output = []
    while t <= tmax:
        k1 = rhs(z, coeffs)
        k2 = rhs(z + 0.5*tstep*k1, coeffs)
        k3 = rhs(z + 0.5*tstep*k2, coeffs)
        k4 = rhs(z + tstep*k3, coeffs)
```

```
        z += tstep/6.0*(k1 + 2*k2 + 2*k3 + k4)
        t += tstep; trec += tstep;
        if trec >= recstep:
            output.append([t,z])
            trec = 0.0
    return output
```

Grafy a experimenty

Nyní otestujeme základní vlastnosti Lotka-Volterra problému. Zvolme $\alpha = 5$, $\beta = 0.06$, $\gamma = 0.02$, $\delta = 5$ a počáteční počty králíků $x(0) = 80$ a lišek $y(0) = 20$.

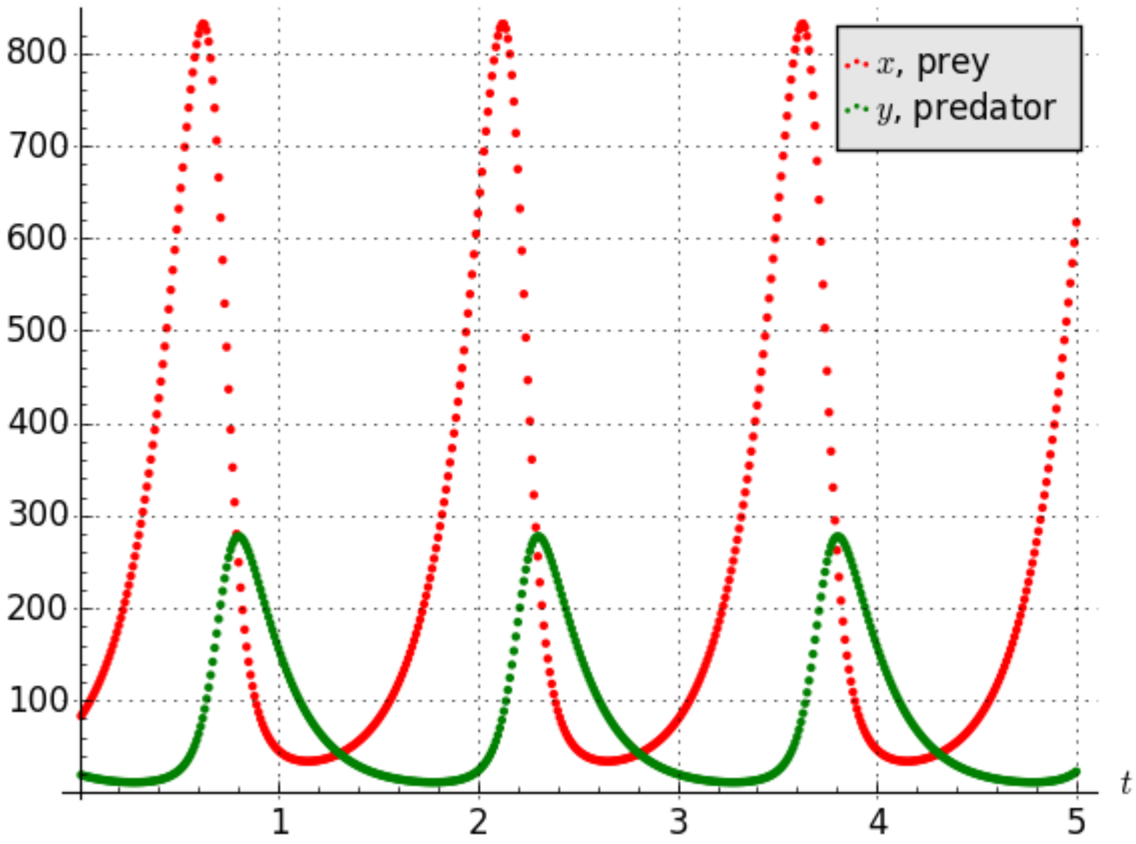
```
data = nsolve([80,20],[5, 0.06, 0.02, 5], 0.001, 5, 0.01)
```

Pro snadnou vizualizaci dat definujeme následující funkce.

```
def getx(ds):
    """ počet králíků v čase """
    return [ [d[0], d[1][0]] for d in ds ]
def gety(ds):
    """ počet lišek v čase """
    return [ [d[0], d[1][1]] for d in ds ]
def phase(ds):
    """ časový vývoj vektoru [králíci, lišky] """
    return [ d[1] for d in ds]
```

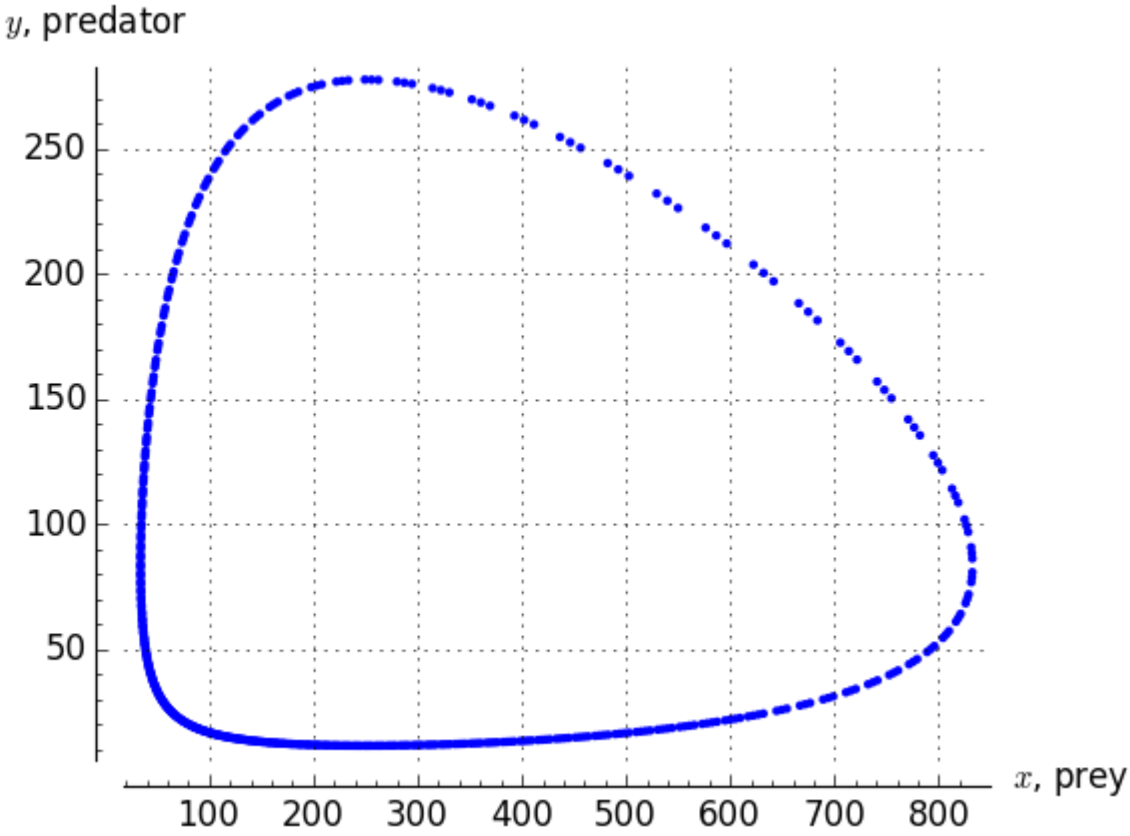
Časový vývoj počtu králíků a lišek.

```
list_plot(getx(data), color='red', legend_label='$x$, prey',
gridlines=true, axes_labels=['$t$',''], figsize=6,
fontsize=12) + list_plot(gety(data), color='green',
legend_label='$y$, predator')
```



Znázornění dynamiky systému v tzv. fázovém prostoru, tedy dvourozměrném prostoru kde jedna souřadnice odpovídá počtu králíků x a druhá počtu lišek y.

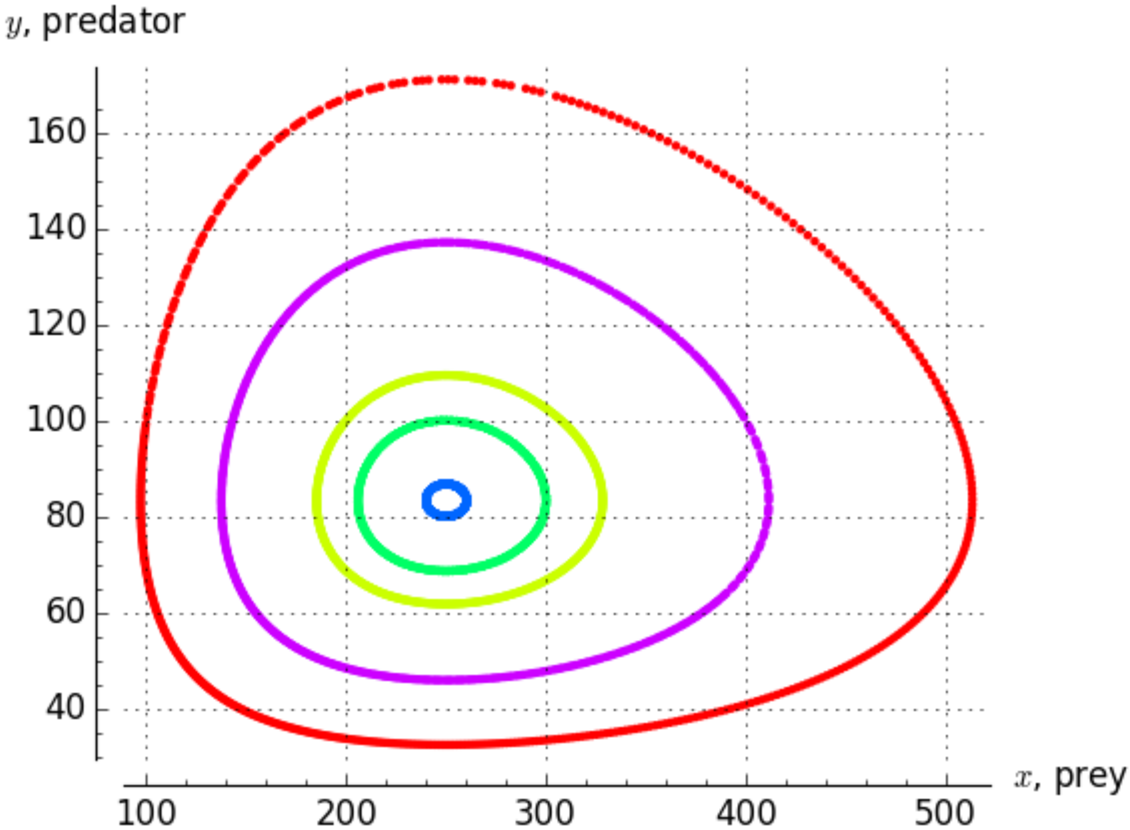
```
list_plot(phase(data), color='blue', gridlines=true,
axes_labels=['$x$, prey', '$y$, predator'], figsize=6,
fontsize=12)
```



Populace samozřejmě oscilují kolem stabilní konfigurace, kterou jsme napočetli v úvodu. Vzpomeňte, že $x_* = \gamma/\delta$ a $y_* = \alpha/\beta$. Můžeme toto chování více zdůraznit vykreslením více trajektorií, pro různé počáteční podmínky.

```
init = [[100,100], [200,100], [250,100], [250,80], [400,100]]
phases = [ phase(nsolve(ic,[5, 0.06, 0.02, 5], 0.001, 5,
0.01)) for ic in init]
```

```
sum([ list_plot(phases[j], color=rainbow(len(phases))[j],
gridlines=true, axes_labels=['$x$, prey', '$y$, predator'],
figsize=6, fontsize=12) for j in range(len(phases)) ])
```



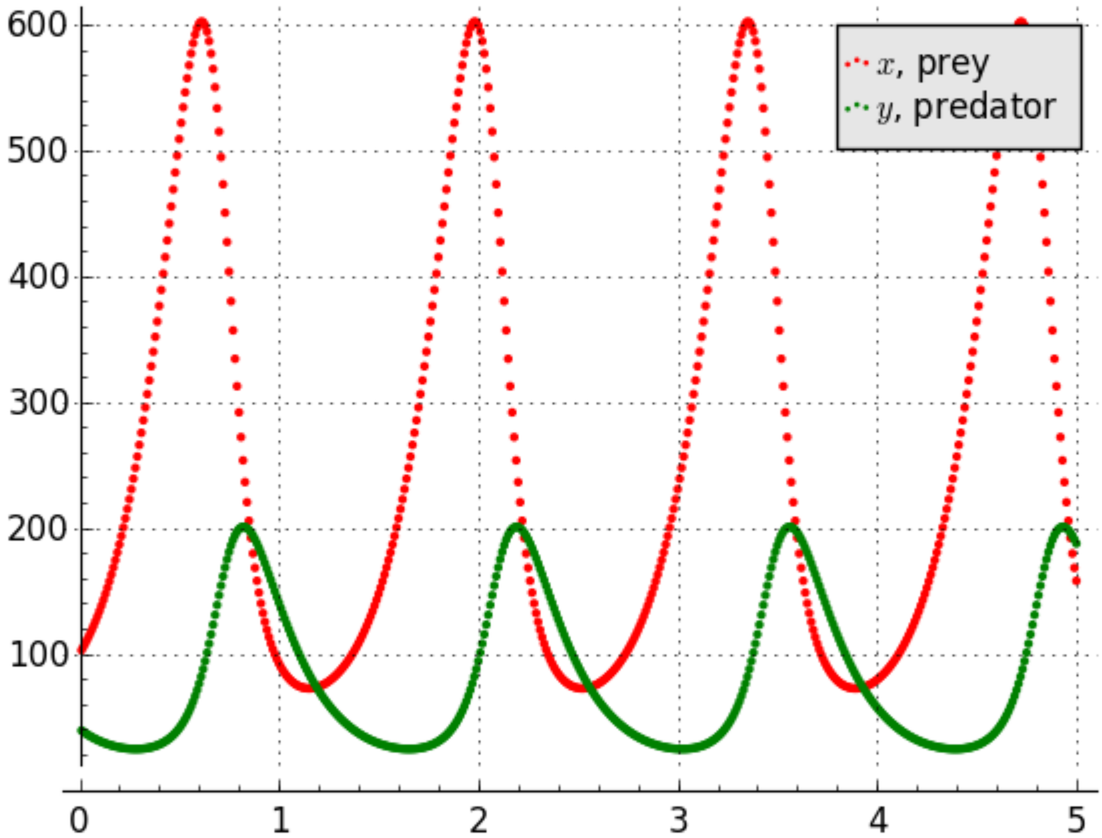
Interaktivní experimenty

Abychom lépe prozkoumali chování systému, rozpohybujeme grafy pomocí interaktivního uživatelského rozhranní.

```
@interact
def _(x=slider(100,500,0.1,label='$x$, prey'),
y=slider(40,160,0.1,label='$y$, predator')):
    data = nsolve([x,y],[5, 0.06, 0.02, 5], 0.001, 5, 0.01)
    figprey = list_plot(getx(data), color='red',
legend_label='$x$, prey', gridlines=true, axes_labels=
['$t$',''], figsize=6, fontsize=12)
    figpred = list_plot(gety(data), color='green',
legend_label='$y$, predator')
    return show(figprey + figpred)
```

x, prey 100.000000000000

y, predator 40.0000000000000



600
500
400
300
200
100
0
1
2
3
4
5
x, prey
y, predator