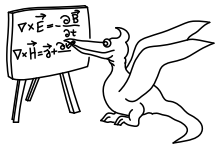


Seriál: Matice a populace

Třicátý první ročník FYKOSu se blíží ke konci a s ním i seriál o numerických metodách a počítačových simulacích. V tomto díle se rozpomeneme na úplný začátek seriálu, kdy jsme klasifikovali simulace, a povíme si něco o populačních modelech. V numerické části se krátce vrátíme k diferenciálním rovnicím, pohovoříme o oblastech stability a nakonec přehledově pojednáme o numerické lineární algebře.

Oblasti stability

V řešení minulého dílu se objevilo tvrzení, že explicitní Eulerova metoda je zcela nevhodná pro řešení netlumeného harmonického oscilátoru a že to vyplývá z její oblasti stability. Pojďme si proto stručně objasnit, o co se jedná. Vezměme si jednoduchou diferenciální rovnici $\dot{y} = \lambda y$, kde λ je obecně komplexní číslo (a tedy i y je komplexní funkce reálné proměnné t). Analytické řešení této rovnice je

$$y(t) = y_0 \mathrm{e}^{\lambda t}\,.$$

Pro $\mathrm{Re}(\lambda) < 0$ řešení s časem klesá k nule, pro $\mathrm{Re}(\lambda) > 0$ naopak exponenciálně roste. Řešme tuto rovnici numericky, pro začátek explicitní Eulerovou metodou. Platí tedy

$$y_{n+1} = y_n + h\lambda y_n = y_n(1 + h\lambda) = y_0(1 + h\lambda)^{n+1}\,. \tag{1}$$

Rozumným požadavkem je, aby numerické řešení nebylo příliš vzdáleno od řešení skutečného, přesněji budeme požadovat, aby pro dané $h\lambda$ bylo řešení omezené[1] (tj. neutíkalo do nekonečna) pro $n \to +\infty$. Tato požadovaná vlastnost se nazývá *absolutní stabilita*. Konkrétně pro explicitní Eulerovu metodu je splněna, pokud $|1 + h\lambda| \leq 1$, jak je vidět z rovnice (1). Body $z = h\lambda$ v komplexní rovině, které splňují podmínku absolutní stability, tvoří tzv. *oblast (absolutní) stability*. Explicitní Eulerova metoda má tedy oblast stability tvořenou kruhem o středu -1 a jednotkovém poloměru. Skutečně, pokud si vzpomenete na diskusi o řešení problémů se silným tlumením v minulém díle, zjistili jsme tehdy, že pro rovnici $\dot{y} = -cy$ musíme volit $h < 2/c$. To přesně odpovídá podmínce na okraj oblasti stability na reálné ose, kde $h\lambda = -hc \geq -2$.

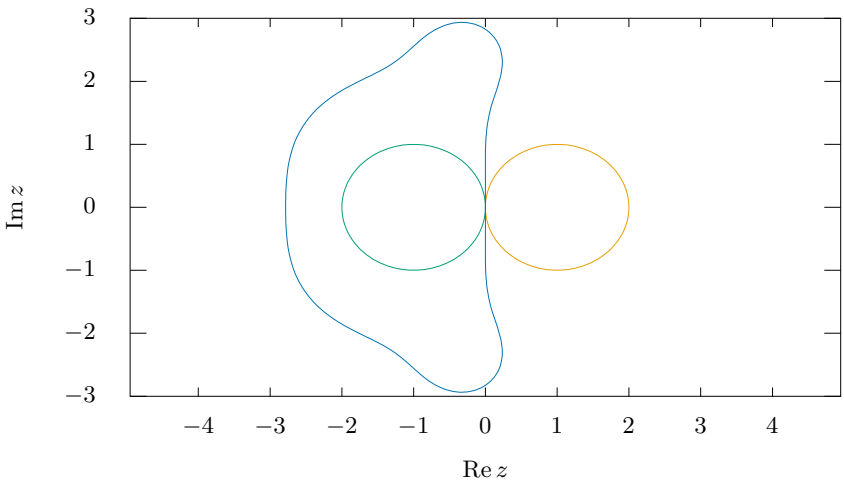
Spočítejme si ještě oblast stability pro implicitní Eulerovu metodu. Z jejího předpisu máme

$$y_{n+1} = y_n + h\lambda y_{n+1} = \frac{y_n}{1 - h\lambda} = \frac{y_0}{(1 - h\lambda)^{n+1}}\,.$$

Aby $|y_n|$ bylo konečné, musí platit $|h\lambda - 1| \geq 1$. Oblast stability implicitní Eulerovy metody je tedy celá komplexní rovina až na jednotkový kruh o středu 1. Oblasti stability pro metody vyšších řádů a pro vícekrokové metody se formálně získávají stejným způsobem, technicky jde ale o náročnější výpočty, proto zde ukážeme pouze výsledek ve formě grafu 1.

Zjistili jsme tedy, jaký musíme volit krok, abychom měli stabilní řešení problému $\dot{y} = \lambda y$. Co ale dělat, pokud máme obecnější problém? Zobecnění mohou být dvou typů. Zaprvé můžeme

[1] Není nyní příliš jasné, proč toto požadujeme i pro $\mathrm{Re}(\lambda) > 0$, kde ani skutečné řešení omezené není. Ukazuje se ale, že tento požadavek i zde nadělá více užitku než škody, protože zamezíme tomu, aby numerické řešení divergovalo ještě rychleji, než skutečné. Hlavní význam této konstrukce je ale nepochybně pro $\mathrm{Re}(\lambda) \leq 0$.



Obr. 1: Hranice oblastí stability pro explicitní (vnitřek levého kruhu) a implicitní (vnějšek pravého kruhu) Eulerovu metodu a Rungovu-Kuttovu metodu 4. řádu (vnitřek „šišoidu“).

mít jednu nelineární rovnici, zadruhé můžeme mít soustavu lineárních rovnic. Nejobecnější je pak kombinace obojího. Pokud máme jednu (skalární) nelineární rovnici ve tvaru $\dot{y} = f(y,t)$, můžeme ji rozvést do prvního řádu[2]

$$f(y,t) \approx f(y_0,t_0) + \left.\frac{\partial f}{\partial y}\right|_{y=y_0} (y-y_0) + \left.\frac{\partial f}{\partial t}\right|_{t=t_0} (t-t_0)\,.$$

Lze přitom ukázat, že na konstantním členu a na členu lineárním v t analýza absolutní stability nezávisí, můžeme tedy pro účely vyšetřování stability aproximovat $\lambda \approx \left.\frac{\partial f}{\partial y}\right|_{y=y_0}$.

Pokud máme lineární soustavu rovnic $\dot{y} = Ay$, kde A je matice a y a $\dot{y}$ jsou vektory, je třeba nalézt tzv. vlastní čísla λ_i matice A a krok h zvolit tak, aby pro všechna tato vlastní čísla platilo, že $h\lambda_i$ leží v oblasti stability. Proč je třeba použít zrovna vlastní čísla je možné nahlédnout například z toho, že pokud s maticí A provedeme operaci zvanou diagonalizace,[3] pak bude mít tato diagonalizovaná matice A' na své hlavní diagonále právě vlastní čísla a jinde budou nuly. Soustava diferenciálních rovnic tedy formálně přejde do tvaru $\frac{\mathrm{d}}{\mathrm{d}t}\tilde{y}_i = \lambda_i \tilde{y}_i$, což již připomíná původně vyšetřovanou rovnici. Značky $\tilde{y}$ zde pouze značí, že jde pouze o formální převod a $\tilde{y}$ neodpovídají původním y.

[2] Jde vlastně o Taylorův rozvoj funkce více proměnných.

[3] ne každá matice je diagonalizovatelná; zde se zaměříme pouze na ty, které jsou – např. všechny symetrické matice

Tento případ si můžeme ilustrovat. Mějme diferenciální rovnici lineárního harmonického oscilátoru $\ddot{y} + \omega^2 y = 0$. Po převodu rovnice na soustavu rovnic prvního řádu máme $\dot{\boldsymbol{y}} = A\boldsymbol{y}$, kde

$$A = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix} .$$

Vlastní čísla této matice[4] jsou $\lambda_{1,2} = \pm \mathrm{i}\omega$, jsou tedy ryze imaginární. Odtud už okamžitě vidíme, že řešení této soustavy explicitní Eulerovou metodou bude pro libovolný krok $h \neq 0$ nestabilní, protože obě $h\lambda_i$ pak neleží v oblasti stability. Naopak např. u Rungovy-Kuttovy metody 4. řádu můžeme volit libovolný krok $h \leq h_{\max} \doteq 2{,}8/\omega$ a metoda bude stabilní.

Poslední, nejsložitější možností je pak kombinace předchozích dvou zobecnění, máme tedy nelineární soustavu rovnic. Jako v prvním případě, i nyní provedeme Taylorův rozvoj do prvního řádu

$$f_i(\boldsymbol{y}, t) \approx f_i(\boldsymbol{y}_0, t_0) + \sum_{j=1}^{N} \left.\frac{\partial f_i}{\partial y_j}\right|_{\boldsymbol{y}=\boldsymbol{y}_0} (y_j - y_{j,0}) + \left.\frac{\partial f_i}{\partial t}\right|_{t=t_0} (t - t_0) .$$

Pokud opět škrtneme všechny členy kromě těch, které jsou lineární v y_j, dostaneme soustavu rovnic $\dot{y}_i = J_{ij} y_j$, kde $J_{ij} = \left.\frac{\partial f_i}{\partial y_j}\right|_{\boldsymbol{y}=\boldsymbol{y}_0}$ jsou složky matice zvané jakobián. Nyní opět nalezneme vlastní čísla matice J_{ij} a aplikujeme stejnou podmínku, jako při zobecnění na soustavu lineárních rovnic.

Tímto bychom ukončili téma řešení obyčejných diferenciálních rovnic. Probrali jsme jej dostatečně podrobně na to, abyste nyní byli schopni numericky vyřešit takřka libovolnou soustavu obyčejných diferenciálních rovnic. Pokud byste se však chtěli dozvědět více, či znovu a jinak slyšet už jednou řečené, podívejte se na záznam přednášky *Počítačové simulace ve fyzice*, která byla přednesena v rámci letošního cyklu FYKOSích přednášek pro středoškoláky a najdete ji na našem Youtube kanálu.[5]

Numerická algebra

Posledním tématem, který si v numerické části seriálu představíme, je numerická lineární algebra. Konkrétně se podíváme na to, jak řešit soustavy lineárních algebraických rovnic. Algebraické rovnice jsou takové ty klasické, které jste znali ze školy jen jako „rovnice“, než jsme vám představili rovnice diferenciální. Lineární jsou, pokud jdou zapsat ve tvaru

$$\sum_{i=1}^{N} a_i x_i = b ,$$

kde a_i, b jsou konstanty a x_i jsou neznámé. Soustavu M takových lineárních rovnic pak můžeme zapsat ve vektorové formě $A\boldsymbol{x} = \boldsymbol{b}$, neboli

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{pmatrix} .$$

[4] To mi v této chvíli musíte věřit.

[5] `https://www.youtube.com/fykosak`

Připomeňme, že maticové násobení funguje s pravidlem „řádek × sloupec“, tedy např. z prvního řádku matice máme rovnici

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1N}x_N = b_1\,.$$

Matici A nazveme *maticí soustavy*, matici

$$(A|b) \stackrel{\text{def}}{=} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2N} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} & b_M \end{pmatrix}$$

pak nazveme *rozšířenou maticí soustavy*. K dalšímu výkladu nyní potřebujeme vysvětlit pojmy lineární nezávislost a řád matice. Řekneme, že množina vektorů $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N\}$ je *lineárně nezávislá*, pokud rovnici $\sum_{i=1}^{N} c_i \boldsymbol{v}_i = \boldsymbol{0}$, kde c_i jsou číselné konstanty, které můžeme zvolit libovolně, splníme pouze tak, že zvolíme všechna $c_i = 0$. V opačném případě jsou vektory *lineárně závislé*. Například vektory $(1,0,2)$, $(2,1,5)$ a $(11,3,25)$ jsou lineárně závislé, neboť

$$5\cdot(1,0,2) + 3\cdot(2,1,5) - 1\cdot(11,3,25) = (0,0,0)\,.$$

V případě matice lze také na jednotlivé sloupce, či řádky pohlížet jako na vektory. *Řád matice* (označme jej rank) je pak maximální počet lineárně nezávislých sloupců (nebo řádků, lze dokázat, že to je jedno) matice. Například matice

$$\begin{pmatrix} 1 & 0 & 2 \\ 2 & 1 & 5 \\ 11 & 3 & 25 \end{pmatrix}$$

má řád 2, neboť všechny tři řádky jsou lineárně závislé. Pokud ale libovolný z nich škrtneme, bude mít výsledná matice už oba dva řádky lineárně nezávislé. Sami si pak můžete ověřit, že to platí i pro její sloupce.

Důvod, proč jsme se zabývali těmito pojmy je ten, že na jejich základě dokážeme určit počet řešení soustavy lineárních rovnic. Pokud je totiž řád rozšířené matice soustavy $(A|b)$ větší než řád matice soustavy A, pak soustava nemá řešení. Naopak,[6] pokud $\operatorname{rank}(A|b) = \operatorname{rank}(A)$, mohou nastat dva případy. Pokud zároveň počet neznámých $N = \operatorname{rank}(A)$, pak má soustava právě jedno řešení. Tímto případem se zde budeme zabývat. Nakonec,[7] pokud $N > \operatorname{rank}(A) = \operatorname{rank}(A|b)$, soustava má nekonečně mnoho řešení.[8] Tyto případy se zpravidla neřeší numericky, proto se jimi nebudeme zabývat.

Možná vás napadá, jestli nejdeme s kanónem na vrabce. Vždyť soustavu lineárních algebraických rovnic zvládneme vyřešit ručně, například Gaussovou eliminací a nepotřebujeme speciální numerické metody. Problém ale nastane, pokud naše soustava rovnic bude velká, metodami, které si představíme lze řešit soustavy až do $N \sim 100\,000$, větší soustavy se nám jednoduše už nevejdou do paměti počítače. Za určitých okolností (matice soustavy je řídká tj. má skoro všechny prvky nulové) lze speciálními metodami řešit i větší soustavy. Jistě uznáte, že takové soustavy bychom už ručně řešili těžko. Takto velké soustavy se v praxi často vyskytují, například při řešení parciálních diferenciálních rovnic někdy nastane krok, kdy získáme obří soustavu

[6] Případ $\operatorname{rank}(A|b) < \operatorname{rank}(A)$ nastat nikdy nemůže.
[7] Případ $N < \operatorname{rank}(A)$ totiž také nemůže nastat.
[8] Libovolná N-tice čísel ale nemusí být řešením, neboť řešení stále musí splňovat onu soustavu rovnic.

lineárních algebraických rovnic, kterou potřebujeme vyřešit. Dalším příkladem může být složitý elektrický obvod řešený pomocí Kirchhoffových zákonů.

Samotné numerické řešení zpravidla probíhá tak, že si matici soustavy rozložíme na součin matic se speciálními vlastnostmi, které nám pak pomohou s maticovou rovnicí lépe pracovat. Správné technické provedení rozkladů je zpravidla složité na vysvětlení, takřka vždy je ale navíc náročné a zdlouhavé na implementaci. V praxi proto (minimálně pro rozklady, spíše však pro celou úlohu) použijte některou z numerických knihoven. My si zde představíme algoritmus pouze pro tzv. LU rozklad, zbytek textu se pak bude zabývat tím, jak nám tyto rozklady pomohou.

LU dekompozice

Nejjednodušším rozkladem, o kterém si zde řekneme, je rozklad matice A na součin horní trojúhelníkové matice U (z angl. upper) a dolní trojúhelníkové matice[9] L (z angl. lower). Horní, resp. dolní trojúhelníková matice je taková, která má nad, resp. pod hlavní diagonálou všechny prvky nulové.

Algoritmus pro LU dekompozici, který si zde představíme, se nazývá Doolittlův a jde v podstatě o Gaussovu eliminaci. Pokud matici A zleva vynásobíme maticí

$$L_1 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -l_{21} & 1 & 0 & \cdots & 0 \\ -l_{31} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -l_{n1} & 0 & 0 & \cdots & 1 \end{pmatrix},$$

kde $l_{k1} = a_{k1}/a_{11}$, dostaneme matici

$$A^{(1)} = L_1 A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{pmatrix},$$

zbavili jsme se tedy v prvním sloupci hodnot pod hlavní diagonálou. Pokud dále zkonstruujeme matice

$$L_k = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -l_{k+1,k} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -l_{n,k} & \cdots & 1 \end{pmatrix},$$

kde $l_{jk} = a_{jk}^{(k-1)}/a_{kk}^{(k-1)}$ pro $j > k$, každou maticí L_k „vyřešíme“ jeden sloupec. Platí tedy ($L_n = I$ vynecháme)

$$L_{n-1} \dots L_2 L_1 A = LA = U,$$

[9] V tomto textu nebude značena L, ale L^{-1}.

kde jsme součin matic L_k označili L. Pokud nyní obě strany vynásobíme inverzní maticí[10] L^{-1}, dostaneme

$$L^{-1}LA = IA = A = L^{-1}U\,,$$

k dokončení LU rozkladu tedy potřebujeme nalézt dolní trojúhleníkovou[11] matici L^{-1}. Inverze matice je obecně výpočetně náročná operace[12], zde ale máme štěstí, neboť lze ukázat, že matice

$$L = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ -l_{21} & 1 & \cdots & 0 \\ -l_{31} & -l_{32} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -l_{n1} & -l_{n2} & \cdots & 1 \end{pmatrix}$$

a její inverze pak je

$$L^{-1} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ l_{31} & l_{32} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix}.$$

Tím máme LU dekompozici zdánlivě zdárně provedenu. Vraťme se nyní k výpočtu l_{jk}. V něm dělíme hodnotami $a_{kk}^{(k-1)}$, které nesmí být nulové, ani kdybychom výpočet prováděli přesně. Při numerickém výpočtu navíc z důvodu redukce zaokrouhlovacích chyb požadujeme, aby tyto hodnoty byly srovnatelné s největšími hodnotami v matici. Řešením je před výpočtem L_k přeházet prvky $A^{(k-1)}$ tak, aby se na k-tém prvku hlavní diagonály vyskytovalo co možná největší číslo. Většinou postačí, pokud prohazujeme řádky, takže hledáme maximum pouze v rámci k-tého sloupce. Pak mluvíme o *částečné pivotizaci*. U opravdu nehezkých matic toto nemusí stačit, pak musíme použít *úplnou pivotizaci*, prohazovat řádky i sloupce a hledat tak maximum v rámci celé matice. Tato prohození si navíc musíme nějakým způsobem pamatovat, neboť jimi vlastně měníme matici, kterou rozkládáme. To je samo o sobě technicky náročnější na implementaci, proto opět doporučujeme použití numerických knihoven. Ovšem i při jejich použití je třeba dát pozor na to, co vrací, neboť většinou provedou rozklad $A = PL^{-1}U$, kde P je právě matice permutací řádků/sloupců vzniklá v důsledku pivotizace. Není tedy možné vzít slepě pouze matice L^{-1} a U vypadlé z metody!

Předpokládejme tedy, že máme úspěšně provedenu dekompozici $A = PL^{-1}U$, řešíme tedy maticovou rovnici $PL^{-1}U\boldsymbol{x} = \boldsymbol{b}$. Výpočet nyní rozložíme do dvou kroků. Nejprve vyřešíme rovnici $PL^{-1}\boldsymbol{y} = \boldsymbol{b}$ pro neznámé $\boldsymbol{y}$. Protože inverze permutační matice je rovna její transpozici, [13] můžeme rovnici přepsat do tvaru $L^{-1}\boldsymbol{y} = P^T\boldsymbol{b}$. Druhým krokem je pak vyřešení rovnice $U\boldsymbol{x} = \boldsymbol{y}$ pro neznámé $\boldsymbol{x}$. Protože L^{-1} i U jsou trojúhelníkové matice, lze maticové rovnice řešit dopřednou a zpětnou substitucí. Tedy například u rovnice $U\boldsymbol{x} = \boldsymbol{y}$ má poslední řádek U pouze jednu nenulovou hodnotu u_{nn}, rovnou tedy určíme $x_n = y_n/u_{nn}$. V předposledním řádku jsou pouze maximálně dvě nenulové hodnoty, z nichž jedna se násobí s již známou hodnotou x_n, rovnou tedy vypočítáme hodnotu x_{n-1} a tak postupujeme dále.

[10] Inverzní matice C^{-1} k matici C je taková, pro kterou platí $C^{-1}C = CC^{-1} = I$, kde I je jednotková matice.

[11] Což ale zatím nevíme, jestli L^{-1} bude splňovat.

[12] Ostatně jedna z cest, jak inverzi numericky efektivně provést, je založena právě na LU dekompozici.

[13] Prohození řádků za sloupce a naopak.

První rovnici lze také vyřešit úpravou do tvaru $\boldsymbol{y} = LP^T\boldsymbol{b}$, musíme si ale dát pozor, abychom násobili v pořadí $L(P^T\boldsymbol{b})$, a ne $(LP^T)\boldsymbol{b}$. V prvním případě totiž z prvního násobení vznikne vektor, provádíme tedy dvě násobení matice a vektoru po sobě, což má složitost $O(n^2)$, zatímco násobení matic ve druhém případě má složitost[14] $O(n^3)$. Možná vás napadne, proč rovnou rovnici neupravíme do tvaru $\boldsymbol{x} = A^{-1}\boldsymbol{b}$ a neprovedeme maticové násobení. Důvod je, jak jsme již poznamenali, že inverze matice je obecně velmi časově náročná.

Ostatní metody a aplikace

Představili jsme si LU dekompozici, což je nejjednodušší z používaných rozkladů matic. Ostatní používané rozklady, jako např. QR či SVD rozklad, jsou složitější a vyžadují znalosti pokročilých pojmů lineární algebry, proto se jimi zde zabývat nebudeme. Tyto rozklady mají mnoho aplikací, kromě již zmíněného řešení soustav lineárních rovnic jde např. o hledání inverze matice, či jejího determinantu. Jak již bylo řečeno, tyto metody jsou (z paměťových důvodů) aplikovatelné pouze na poměrně malé[15] matice, často je ale potřeba řešit daleko větší soustavy rovnic. Pokud máme štěstí, a tato matice je řídká, tj. má skoro všechny prvky nulové, je možné si zapamatovat pouze kde se nachází tyto nenulové prvky a jaké jsou jejich hodnoty, čímž docílíme značné paměťové úspory. S takto reprezentovanou maticí ale zpravidla nedokážeme efektivně provádět libovolnou operaci, speciálně nechceme provádět rozklad na součin matic, neboť tyto matice by již nemusely být řídké. V takovém případě pak používáme různé iterační metody, tedy vyjdeme z nějakého odhadu $\boldsymbol{x}^{(0)}$, na který opakovaně aplikujeme algoritmus metody a určujeme přesnější odhady $\boldsymbol{x}^{(k)}$ dokud nedokonvergujeme s dostatečnou přesností k výsledku. Tyto metody pro řešení řídkých soustav opět najdete v numerických knihovnách.

Zajímavou aplikací je taktéž fitování (experimentálních) dat polynomiální závislostí. Pokud totiž máme m naměřených dat (x_i, y_i) a teoretickou závislost $y(x) = b_0 + \sum_{j=1}^{n} b_j x^j$, máme soustavu rovnic

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^n \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{pmatrix},$$

neboli $\boldsymbol{y} = X\boldsymbol{b} + \boldsymbol{\epsilon}$, kde $\boldsymbol{b}$ jsou hledané regresní koeficienty a $\boldsymbol{\epsilon}$ jsou skutečné experimentální chyby. Protože naměřených hodnot je víc než hledaných koeficientů, řešíme přeurčenou soustavu, která pro $\boldsymbol{\epsilon} = 0$ nemá řešení. Ostatně když provedeme regresi, závislost nám nikdy přesně neprotíná všechny experimentální body, ty jsou místo toho rozmístěny okolo regresní závislosti. Místo přesného řešení se tedy snažíme najít takové koeficienty, pro které budeme řešení v jistém smyslu nejblíže, snažíme se tedy minimalizovat velikost vektoru $||\boldsymbol{\epsilon}||$. Takový problém lze[16] převést na (nepřeurčenou) soustavu lineárních rovnic pro $\boldsymbol{b}$. Vidíme tedy, že na aplikace numerické lineární algebry narazíme téměř na každém kroku.

[14] Existují pokročilé algoritmy, které to zvládnou o chlup rychleji.
[15] $n = 10\,000$ je malá matice, jak se přesvědčíte v úloze.
[16] Postup ale vyžaduje znalost pokročilé matematiky, nebudeme jej tedy zde rozebírat.

Řešení Poissonovy rovnice

V již odkazované přednášce pro středoškoláky jsme vyřešili Laplaceovu rovnici ve 2D pomocí Excelu. Vaší úlohou bude totéž[17] provést v Pythonu. Pojďme si proto zrekapitulovat potřebnou teorii.

Jedna z Maxwellových rovnic má podobu $\operatorname{div} \boldsymbol{E} = \varrho/\varepsilon_0$, kde $\boldsymbol{E}$ je intenzita elektrického pole, ϱ je hustota náboje a ε_0 je permitivita vakua. Pokud do ní dosadíme definici elektrického potenciálu $\boldsymbol{E} = -\operatorname{grad}\varphi$, obdržíme Poissonovu rovnici

$$\Delta\varphi = -\frac{\varrho}{\varepsilon_0}\,.$$

V případě, že je všude $\varrho = 0$, dostaneme tzv. Laplaceovu rovnici

$$\Delta\varphi = 0\,.$$

Pokud tedy dostaneme zadanou hustotu náboje ve všech bodech oblasti a navíc tzv. okrajové podmínky, řešením této rovnice obdržíme potenciál φ ve všech bodech prostoru.

Poissonova rovnice je ukázkou parciální diferenciální rovnice, neboť Laplaceův operátor (zde ve 2D) $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ obsahuje parciální derivace. Potenciál $\varphi(x, y)$ je totiž funkcí více proměnných, musíme tedy specifikovat, podle které zrovna derivujeme. Všechny ostatní v tu chvíli (pokud derivujeme parciálně) považujeme za konstantu.

Poissonova rovnice je PDR takového typu, ve které nevystupuje derivace podle času. Nehledáme tedy časový vývoj, ale stacionární řešení. Z tohoto důvodu také nespecifikujeme počáteční podmínku, jako u ODR, musíme ale specifikovat tzv. okrajové podmínky. Rovnici totiž řešíme na nějaké konečné oblasti, derivace ale vyžadují znalost hodnot v okolí daného bodu. Na hranici této oblasti tedy musíme předepsat dodatečné informace, kterým se říká okrajové podmínky.

Nejjednodušší okrajovou podmínkou je *Dirichletova podmínka*, která je ve tvaru $\varphi(x, y) = 0$, resp. obecněji $\varphi(x, y) = C(x, y)$, kde C je pevně daná (většinou konstantní) funkce. Dirichletova podmínka tedy říká, jaká je hodnota potenciálu v daném bodě hranice. Druhou používanou podmínkou je *von Neumannova podmínka*, která je ve tvaru $\frac{\partial\varphi}{\partial x} = 0$, kdy derivaci provádíme vždy ve směru kolmo k hranici. Tato podmínka říká, že v daném bodě se hodnota potenciálu v daném směru nemění (určuje vlastně hodnotu elektrického pole).

Platí přitom, že v každém bodě hranice stačí specifikovat jednu z těchto dvou podmínek, aby bylo řešení jednoznačné. Dirichletovu podmínku použijeme, pokud víme hodnotu potenciálu, například na elektrodách kondenzátoru. Von Neumannovu podmínku pak použijeme, pokud jsme na hranici, kde předpokládáme v daném směru homogenní řešení. Použijeme je tedy např. ve volném prostoru daleko od všech zdrojů. Okrajové podmínky přitom nemusí být specifikovány pouze na samém okraji oblasti, ale kdekoliv, kde tyto podmínky potřebujeme specifikovat (a v těchto bodech pak neřešíme samotnou PDR). Pokud tedy např. do deskového kondenzátoru vložíme další elektrodu, specifikujeme ji pomocí Dirichletovy podmínky v oblasti, kde se elektroda nachází.

Nyní si povíme, jak Poissonovu rovnici vyřešit numericky. Použijeme k tomu metodu konečných diferencí, kdy budeme hledat řešení na nějaké (v našem případě pravoúhlé a pravidelné) mřížce. Laplaceův operátor obsahuje druhé derivace ve směrech x a y, pokud tedy najdeme numerickou aproximaci druhé derivace, najdeme i numerickou aproximaci Laplaceova operátoru.

[17] Excel vnitřně používal nějakou z iteračních metod, zatímco my budeme soustavu rovnic řešit přesně.

Obdobným postupem, jako v kapitole o numerické derivaci si napišme dva Taylory a sečtěme je. Dostaneme

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f^{(3)}(x) + O(h^4)\,,$$
$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f^{(3)}(x) + O(h^4)\,,$$
$$f(x+h) + f(x-h) = 2f(x) + h^2 f''(x) + O(h^4)\,.$$

Druhou derivaci tedy můžeme aproximovat vztahem

$$f_i'' = \frac{f_{i+1} + f_{i-1} - 2f_i}{h^2} + O(h^2)\,,$$

kde h je vzdálenost dvou sousedních bodů mřížky. Za předpokladu, že krok h je stejný ve směru x i y, tedy můžeme pro Laplaceův operátor v bodě (i,j) mřížky, kde má funkce $f(x,y)$ hodnotu $f_{i,j}$, psát

$$\Delta f_{i,j} = \frac{f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1} - 4f_{i,j}}{h^2} + O(h^2)\,.$$

Tímto problém převedeme na řešení soustavy rovnic $A\boldsymbol{\phi} = \boldsymbol{b}$, kde matice A na jednotlivých řádcích (odpovídajících rovnicím pro jednotlivé body mřížky) obsahuje buď aproximaci Laplaceova operátoru výše, nebo některou z okrajových podmínek. Vektor $\boldsymbol{b}$ pak obsahuje hodnoty $-\varrho/\varepsilon_0$ v jednotlivých bodech mřížky, případně pravé strany okrajových podmínek. Řešením této soustavy pak získáme hodnoty potenciálu $\boldsymbol{\phi}$ ve všech bodech mřížky.

Populační modely

Prakticky ve všech úlohách, které jsme v předešlých dílech seriálu řešili, jsme se věnovali pouze diskrétním simulacím. Nyní využijeme nabyté znalosti z minula a podíváme se na spojité simulace, tj. takové, které vyžadují numerické řešení diferenciálních rovnic.

Diferenciální rovnice mají nepřeberné množství aplikací. Abychom si trochu odpočinuli od fyziky a podporovali mezioborové vzdělávání, zaměříme se na (v biologii hojně používané) modely vývoje populace.[18]

Základní populační modely

Není potřeba dlouze zmiňovat, že časový vývoj velikosti populace je ovlivněn širokou paletou různých faktorů a není tedy možné jeho dynamiku exaktně postihnout. O to se ani nebudeme pokoušet. Místo toho se zaměříme jen na pár důležitých vlivů a budeme sledovat, co se děje s populací, když má navrch ten či onen faktor.

Nejjednodušší je *exponenciální růstový model*. Ten je založen na předpokladu, že produkce potomků není nijak regulována, tj. prostředí poskytuje neomezený přísun potravy, prostoru a nežijí v něm predátoři. Potom můžeme vývoj počtu jedinců $n(t)$ zapsat pomocí diferenciální rovnice

$$\frac{\mathrm{d}n}{\mathrm{d}t} = rn\,. \tag{2}$$

[18] V principu nám nic nebrání použít diskrétní simulace (diferenční rovnice), ale jelikož se děti/mláďata nerodí skokově, ale neustále a nepravidelně, dává lepší smysl použít diferenciální rovnice.

Tato rovnice má jednoduché řešení

$$n(t) = n(t_0)\mathrm{e}^{r(t-t_0)}\,. \tag{3}$$

Parametr r se nazývá *růstový koeficient*, t_0 je počáteční čas. Často budeme pokládat $t_0 = 0$ a psát $n_0 \equiv n(t_0 = 0)$.

Pro $r > 0$ populace roste, pro $r < 0$ asymptoticky klesá k nule, pro $r = 0$ vývoj stagnuje (populace je konstantní). Pro praktické účely je potřeba buďto koeficient extrapolovat ze starších dat, nebo mít teorii pro jeho odhad.[19] S exponenciálním růstem či poklesem se samozřejmě setkáváme i ve fyzice, příkladem budiž ubývání částic radioaktivního nuklidu v důsledku rozpadu na lehčí částice – viz minulý díl seriálu, první kapitola.

Za jeden z prvních příkladů aplikace exponenciálního modelu v biologii můžeme považovat Fibonacciho úlohu s králíky. Fibonacci uvažoval model, v němž máme na počátku dva mladé králíky, přičemž mladý králík se stane dospělým za jeden měsíc (jednotka času, není podstatná pro teorii), a dospělý pár má každý měsíc dva nové potomky (pro šťouraly: nechť je v každém páru jeden samec a jedna samice). Počet všech párů králíků v jednotlivých časových krocích je v takovém případě dán sekvencí $1, 1, 2, 3, 5, 8, 13, \ldots$ Posloupnost, jejíž každý člen je dán součtem dvou předchozích, je na počest autora nazývána Fibonacciho. Pro nás je podstatné, že tento diskrétní model se chová v limitě velkých časů podle vztahu $n(t) = \varphi^t/\sqrt{5}$, kde $\varphi = (1+\sqrt{5})/2$ je zlatý řez, což volbou $r = \ln\varphi$ a $n_0 = 1/\sqrt{5}$ převedeme na rovnici (3).

Druhým významným populačním modelem, který je zároveň stále jednoduchý, je *logistický růst*. Je vyjádřen diferenciální rovnicí

$$\frac{\mathrm{d}n}{\mathrm{d}t} = rn\left(1 - \frac{n}{k}\right)\,. \tag{4}$$

Koeficient k zde představuje tzv. *nosnou kapacitu prostředí*. Pokud bude kapacita nekonečná, hodnota závorky bude 1 a dostaneme zpět neregulovaný, exponenciální růst. Na hodnotu v závorce lze pohlížet jako na faktor, který reguluje velikost růstového koeficientu r. Pokud je velikost populace významně menší než je kapacita prostředí, $n \ll k$, je hodnota v závorce zhruba rovna 1 a dostáváme tak zmíněný neregulovaný růst. Jak se poměr n/k blíží zdola jedné, růst postupně zpomaluje, až se velikost populace zastaví na hodnotě $n = k$; prostředí je saturováno.

S tímto chování se opět setkáváme i ve fyzice, a to u fermionového plynu. Zatímco ideální plyn složený z klasických částic lze komprimovat do libovolně malého objemu, u fermionů (např. elektrony) tomu tak není. Pauliho vylučovací princip totiž říká, že žádný kvantový stav nemůže být okupován dvěma či více fermiony s identickou sadou kvantových čísel. Proto může dojít k nasycení kvantových stavů, což má za následek například zastavení kolapsu hvězdy po dohoření paliva a následný vznik bílého trpaslíka.[20]

Jelikož je tento model mírně komplikovanější než předchozí, rozepíšeme zde postup řešení a výsledek rozebereme. Rovnici (4) separujeme a budeme integrovat,

$$\int_{n_0}^{n} \frac{\mathrm{d}n}{n\left(1-\frac{n}{k}\right)} = \int_0^t r\mathrm{d}t\,.$$

[19] Takový teoretický vztah může obsahovat parametry jako počet potomků v jednom vrhu, dobu mezi vrhy, délku života atp.

[20] Jedná se ovšem o aproximaci. Tlak elektronového plynu může být překonán, což vede ke vzniku neutronové hvězdy nebo dokonce černé díry.

Na pravé straně dostáváme triviálně rt. Na levé straně použijeme substituci $n = n'k$ a provedeme rozklad na parciální zlomky

$$\frac{1}{n'(1-n')} = \frac{1}{n'} - \frac{1}{n'-1}\,.$$

Tyto zlomky se integrují na logaritmy, a tak vrácením substituce a převedením rozdílu logaritmů na logaritmus podílu postupně dostaneme

$$\ln\left(\frac{n'}{n'_0}\right) - \ln\left(\frac{1-n'}{1-n'_0}\right) = \ln\left(\frac{n}{n_0}\right) - \ln\left(\frac{1-\frac{n}{k}}{1-\frac{n_0}{k}}\right) = \ln\left(\frac{n}{n_0}\,\frac{1-\frac{n_0}{k}}{1-\frac{n}{k}}\right)\,.$$

Výsledek

$$\ln\left(\frac{n}{n_0}\,\frac{1-\frac{n_0}{k}}{1-\frac{n}{k}}\right) = rt$$

převedeme do exponenciálního tvaru

$$\frac{n}{1-\frac{n}{k}} = c\mathrm{e}^{rt}\,,$$

kde jsme použili substituci $c = n_0/(1-n_0/k)$. Jelikož se jedná o rovnici lineární v n, můžeme snadno vyjádřit

$$n(t) = \frac{k}{1+\frac{k}{c}\mathrm{e}^{-rt}}\,. \tag{5}$$

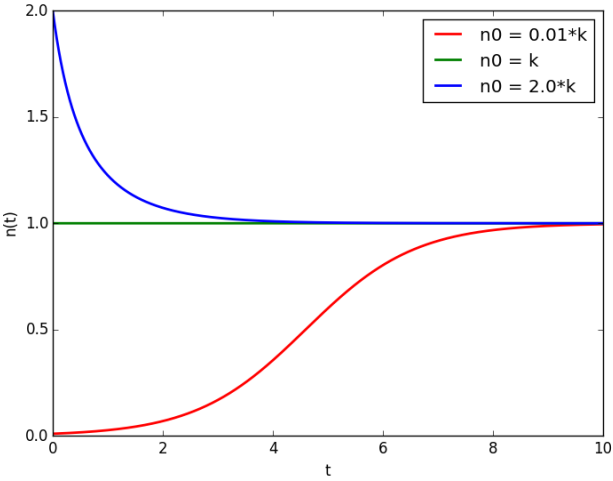
Funkce (5) se nazývá *logistická funkce*. Na základě hodnoty parametru c rozlišujeme tři kvalitativně odlišná řešení. Pro $n_0 = k$ je $c = \infty$ a funkce se tak redukuje na $n = k$, jak jsme si všimli výše. Pro $n_0 < k$ je $c > 0$, z čehož plyne $n(t) < k\,\forall t$, přičemž díky členu e^{-rt} máme limitu $\lim_{t\to\infty} n(t) = k$. Třetím řešením je zavedení přesycené populace $n_0 > k$. Potom máme $c < 0$, což má za důsledek $n(t) > k\,\forall t$ a limita nekonečného času je opět k. Populace, která je větší než nosná kapacita prostředí, postupně konverguje k této kapacitě. Všechna tři řešení naleznete na obrázku 2. Ještě uveďme, že model lze dále vylepšit například zavedením časové závislosti nosné kapacity či přičtením fluktuačního členu, řešení se pak ovšem může podstatně zkomplikovat.

Lotkův–Volterrův model

Oba výše zmíněné modely předpokládaly, že se ve zkoumaném prostředí vyskytuje pouze jedna populace. Takový případ můžeme zřídka pozorovat např. u některých autotrofů–extremofilů, typicky ovšem pozorujeme dvě a více populací, které spolu interagují. V následujících odstavcích zmíníme dva typy interakce a rozebereme modely, kterými je můžeme popsat.

Prvním typem je interakce predátor–kořist. Kořistí je nejčastěji býložravec, například již zmíněný králík, který získává potravu (trávu) z prostředí. Budeme uvažovat, že trávy je neomezeně mnoho, populace králíků se tedy sama o sobě bude vyvíjet exponenciálně. Predátor, například liška, se živí lovem králíků. Čím větší je populace králíků, tím rychleji roste populace lišek. Naopak čím větší je populace lišek, tím pomaleji roste (rychleji klesá) populace králíků.

Již z uvedeného popisu si můžeme rozmyslet, že populace budou nejspíše oscilovat, přičemž zde bude určitý fázový posun, neboť když je jedna populace na maximu, nemůže být v maximu



Obr. 2: Logistický vývoj populace pro různé počáteční hodnoty. Zelená křivka představuje případ, kdy je populace již od začátku na nosné kapacitě, červená křivka odpovídá růstu z hodnot zanedbatelných vůči kapacitě a modrá křivka je vývoj populace nad nosnou kapacitou prostředí. Pro jednoduchost jsme volili $k = 1$.

i druhá. Matematický model vývoje populací se nazývá *Lotkův–Volterrův model predátor–kořist* a je shrnut soustavou diferenciálních rovnic

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= r_{\mathrm{x}}x - D_{\mathrm{x}}xy\,, \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= r_{\mathrm{y}}xy - D_{\mathrm{y}}y\,. \end{aligned} \tag{6}$$

Koeficienty r_{x}, r_{y} jsou růstové, koeficienty D_{x}, D_{y} jsou extinkční.[21] Všechny tyto koeficienty jsou kladné.

Rovnice dobře postihují fakt, že zatímco predátora y interakce (člen xy) živí, kořist zase hubí. Špatnou zprávou je, že tyto rovnice nám neumožňují najít časovou závislost velikosti jednotlivých populací v uzavřeném tvaru. Soustavu nelineárních diferenciálních rovnic (6) je možné linearizovat, dostaneme tak ale pouze aproximativní řešení pro malé relativní změny v populacích, což není uspokojivé. Exaktně dokážeme nalézt například stacionární bod, tj. velikosti populací predátora a kořisti, při kterých dojde ke stagnaci. Stačí prostě položit $\dot{x} = 0$, $\dot{y} = 0$ a vyřešit soustavu dvou lineárních algebraických rovnic. Výsledkem jsou kořeny $(x_1, y_1) = (0, 0)$ a $(x_2, y_2) = (D_{\mathrm{y}}/r_{\mathrm{y}}, r_{\mathrm{x}}/D_{\mathrm{x}})$. Lze ukázat, že bod (x_1, y_1) je sedlový, a tedy nestabilní, a tedy že populace nemají tendenci vymřít, což ukazuje na kvalitu modelu. Perturbace okolo (x_2, y_2) vedou k oscilacím ve vývoji populace (zmíněné linearizované řešení). Vyšetřování stability ře-

[21] Koeficienty v Lotkově–Volterrově modelu se často uvádějí bez konkrétních názvů, vybrali jsme jim proto nějaké příhodné.

šení diferenciálních rovnic je velice zajímavá disciplína, ale nemáme zde na ni ani prostor, ani matematický aparát.

Je tu ale i dobrá zpráva, a sice, že pokud nám bude stačit sledovat vývoj pouze během několika mála period, vystačíme si při integraci zcela bezpečně s oblíbenu explicitní metodou RK4. Níže je přiložen kód, který řeší Lotkovy-Volterrovy rovnice. Získaný vývoj je vykreslen na obrázku (3). V obrázku si všimněte stavu, kdy se populace kořisti přiblíží k nulové hodnotě. Při aplikaci modelu v biologii musíme mít na paměti, že aby se kořist mohla rozmnožovat, musí přežít alespoň jeden jedinec. Pokud bychom hodnotu x považovali za počet jedinců, je z obrázku 3 zřejmé, že pro zvolené parametry model selhal. Tento jev se v literatuře označuje jako „atto-fox problem“ (při modelování vývoje populací králíků a lišek v Británii klesla populace predátorů na jednu attolišku, tj. 10^{-18} lišky).

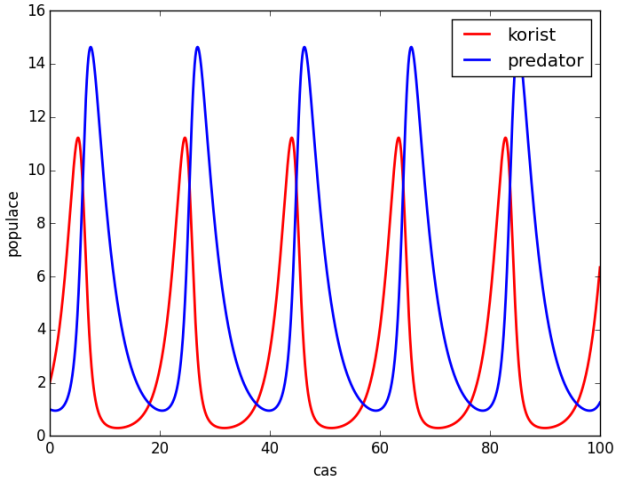
```
# LOTKUV-VOLTERRUV MODEL PREDATOR KORIST
# Importujeme knihovny
import numpy as np
from matplotlib import pyplot as plt

# metoda Runge-Kutta 4. radu pro funkce bez explicitni casove zavislosti
def rk4(f, y, h):
    k1 = f(y)
    k2 = f(y + (h/2.)*k1)
    k3 = f(y + (h/2.)*k2)
    k4 = f(y + h*k3)
    y = y + (h/6.)*(k1 + 2.*k2 + 2.*k3 + k4)
    return y

# Lotkuv-Volterruv model, r rust, D extinkce,
# prvni prvek pole korist, druhy predator
def lv_pp(y):
    r = np.array([0.5,0.1])
    D = np.array([0.1,0.3])
    return np.array([r[0]*y[0] - D[0]*y[0]*y[1],r[1]*y[0]*y[1] - D[1]*y[1]])

# definujeme delku casoveho kroku, pocet kroku, pocatecni velikost populaci
h = 0.01
kroky_max = 9999
krok = 0
cas_pole = np.zeros(kroky_max+1)
korist_pole = np.zeros(kroky_max+1)
predator_pole = np.zeros(kroky_max+1)
y = np.array([2.,1.])
korist_pole[krok] = y[0]
predator_pole[krok] = y[1]
# integrujeme pomoci rk4, ukladame do pole
while krok < kroky_max:
    y = rk4(lv_pp,y,h)
    krok += 1
    korist_pole[krok] = y[0]
    predator_pole[krok] = y[1]
    cas_pole[krok] = krok*h

# vykreslim casovou zavislost
koristplot, = plt.plot(cas_pole,korist_pole,'r',linewidth=2,label='korist')
predatorplot, = plt.plot(cas_pole,predator_pole,'b',linewidth=2,label='predator')
plt.xlabel('cas')
plt.ylabel('populace')
plt.legend(handles=[koristplot, predatorplot])
plt.show()
```



Obr. 3: Vývoj populací podle Lotkova–Volterrova modelu predátor–kořist. Červená čára představuje kořist, modrá predátora. Volili jsme koeficienty $r_\mathrm{x} = 0{,}5$, $r_\mathrm{y} = 0{,}1$, $D_\mathrm{x} = 0{,}1$, $D_\mathrm{y} = 0{,}3$ a počáteční populace $x = 2$, $y = 1$. Velikost populace považujeme za normalizovanou, tzn. nepředstavuje počet jedinců.

Další a poslední populační model, který v tomto textu rozebereme, je *kompetitivní Lotkův–Volterrův model*. Zatímco model predátor–kořist přidával interakci k exponenciálnímu modelu, kompetitivní model přidává interakci k logistickému modelu. Příslušná soustava diferenciálních rovnic má pro dvě populace tvar

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= r_\mathrm{x} x \left(1 - \left(\frac{x + I_\mathrm{xy} y}{k_\mathrm{x}}\right)\right), \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= r_\mathrm{y} y \left(1 - \left(\frac{y + I_\mathrm{yx} x}{k_\mathrm{y}}\right)\right). \end{aligned} \tag{7}$$

V tomto modelu nedochází k predaci, populace však soupeří o společný zdroj potravy, což je vyjádřeno členy $I_\mathrm{xy} y$ a $I_\mathrm{yx} x$, kde interakční koeficienty I jsou kladné. Na závorku se opět můžeme dívat jako na parametr upravující růstový koeficient, interakční koeficienty pak můžeme interpretovat jako poměr, který nám říká, kolika jedincům v populaci x zabere životní prostor jedinec z populace y.

Podobně jako v případě modelu predátor–kořist neexistuje analytické řešení a musíme se spolehnout na numeriku. Co se týče stacionárních bodů, řešíme soustavu dvou kvadratických

rovnic a nalezneme kořeny

$$\begin{aligned} &x_1 = 0\,, y_1 = 0\,,\\ &x_2 = 0\,, y_2 = k_\mathrm{y}\,,\\ &x_3 = k_\mathrm{x}\,, y_3 = 0\,,\\ &x_4 = \frac{k_\mathrm{x} - I_\mathrm{xy}k_\mathrm{y}}{1 - I_\mathrm{xy}I_\mathrm{yx}}\,, y_4 = \frac{k_\mathrm{y} - I_\mathrm{yx}k_\mathrm{x}}{1 - I_\mathrm{xy}I_\mathrm{yx}}\,. \end{aligned}$$

Zde by byl další rozbor podstatně komplikovanější než u modelu s predací. Pouze si proto povšimněme, že čtvrtý kořen[22] může v případě silných interakcí nabývat záporných hodnot, tj. v kladné oblasti parametrů x, y (záporné nemají v biologii smysl) neexistuje žádný bod, okolo kterého by mohlo řešení oscilovat, nebo k němu konvergovat.

Pro řešení kompetitivního Lotkova–Volterrova modelu lze opět využít metodu RK4. Studování dynamiky modelu bude vaším úkolem v seriálové úloze.

Na závěr ještě dodejme, že Lotkovy–Volterrovy diferenciální rovnice lze zobecnit a psát jako soustavu[23]

$$\frac{\mathrm{d}n_i}{\mathrm{d}t} = n_i f_i(\boldsymbol{n})\,,$$

kde

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{r} + A\boldsymbol{x}\,.$$

Složka n_i vektoru $\boldsymbol{n}$ udává velikost i-té populace. Složky r_i vektoru $\boldsymbol{r}$ jsou růstové koeficienty jednotlivých populací a matice A popisuje (potenciálně nesymetrické) interakce mezi populacemi. V tomto tvaru můžeme tedy do modelu zavést libovolný počet populací s různými typy interakcí. Například můžeme vyjít z kompetitivního modelu a některým populacím dát záporné interakční koeficienty, čímž budeme simulovat parazitické chování (pokud bude $I_{ij} < 0$ a zároveň $I_{ji} < 0$, půjde o mutualismus), a k tomu přidat populace, které se živí jen predací těch, na nichž jsou závislí paraziti. Tak získáme nepřímou interakci mezi predátory a parazity.

Co se nevešlo

V průběhu roku jsme vám v seriálu slíbili probrat několik témat, která jsme později jen stručně odbyli, nebo se na ně vůbec nedostalo. Bohužel jsme s tím nemohli nic udělat, protože už v současném stavu je tento seriál zdaleka nejdelší v historii FYKOSu, a díky častým požadavkům na psaní vlastních počítačových skriptů je také jedním z nejnáročnějších seriálů, které jsme kdy mezi naše řešitele vypustili. Pokusíme se zde proto vysvětlit, co nás vedlo k vynechání těch konkrétních témat.

V simulační části zcela chybělo pojednání o termodynamickém Monte Carlu (TMC). Tato hojně využívaná metoda by vyžadovala shrnout základy termodynamiky a statistické fyziky, s nimiž se potýkal seriál 29. ročníku, a ještě je trochu rozšířit. A tím jsme vás nechtěli trápit. Zároveň jsme vás proto museli ochudit o rozbor Isingova modelu ferromagnetismu, který se

[22] všimněte si, že explicitně neobsahuje nulu (vyhynulou populaci) – pokud $k_\mathrm{x} - I_\mathrm{xy}k_\mathrm{y} = k_\mathrm{y} - I_\mathrm{yx}k_\mathrm{x} = 0$, je i jmenovatel $1 - I_\mathrm{xy}I_\mathrm{yx} = 0$; dosazením do (7) zjistíme, že pak je stabilní libovolná populace s $x/k_\mathrm{x} + y/k_\mathrm{y} = 1$

[23] Pro ty, kteří již slyšeli o Einsteinově sumační konvenci, pro jistotu uvádíme, že zde se přes stejné indexy nesčítá.

nejlépe řeší právě pomocí TMC.[24] Alternativní přístupy pomocí buněčných automatů bohužel nejsou na pochopení o moc jednodušší. Abyste si nemysleli, že Isingův model složí jenom k modelování ferromagnetismu, tak vězte, že pomocí něj lze předpovídat i úrodu pistácií.[25]

Stejně tak jsme se z důvodů zvýšené složitosti a protahování délky textu nevěnovali simulacím kritických jevů pomocí buněčných automatů a jevu perkolace. Také jsme vynechali poměrně zábavné simulace přírodních katastrof. Více času jsme také mohli věnovat studiu Brownova pohybu. Celkově lze říct, že prakticky na každém tématu, kterého jsme se během simulací dotkli, si lze založit vědeckou kariéru. Pokud to ovšem člověk s modelováním přírodních jevů myslí vážně, je potřeba strávit pár let studiem matematiky a statistiky. Jak jsme již dříve zmínili, statistické zpracování simulací jsme v seriálu rychle vzdali, neboť činilo řešitelům problémy a ubíralo prostor přitažlivějším tématům. Není však pravda, že by nebylo bez hlubokých matematických znalostí něco pěkného nasimulovat – to jsme se ostatně snažili v tomto textu dokázat. Pokud máte chuť se věnovat simulacím i bez toho, abyste byli vedeni FYKOSem, zkuste se podívat třeba na evoluční teorii her.[26] Pokud jste trochu více pragmatičtí, můžete si zkusit nasimulovat některé jevy v ekonomii, zkuste hledat heslo *Black-Scholes equations*. Pokud se vám líbil aktuální biologicky zaměřený díl, zkuste vyhledat difúzní populační modely či populační modely založené na celulárních automatech. Možností je spoustu, stačí si jen vybrat.

Poděkování

Chtěli bychom vyjádřit velký dík vám všem, kteří jste dočetli až sem, a zároveň vám poblahopřát k vašemu neochvějnému zájmu o počítačovou fyziku. Během psaní seriálu jsme seznali, že je pro většinu řešitelů dosti náročný, což se negativně projevilo na počtu odevzdaných řešení. Ačkoli jsme se postupně snažili psát text více srozumitelně a s menším množstvím matematiky, rostoucí nároky na samostatné programování byly bohužel pro mnohé FYKOSáky velkou překážkou, takže vás do konce vydržela jen hrstka. Avšak o to víc na sebe můžete být hrdí, a vězte, že nabyté programovací dovednosti v životě ještě mnohokrát zúročíte. Mnoho zdaru s šestou sérií a neutuchající nadšení pro fyziku vám přejí

Mirek Hanzelka a Lukáš Timko

Fyzikální korespondenční seminář je organizován studenty MFF UK. Je zastřešen Oddělením propagace a mediální komunikace MFF UK a podporován Ústavem teoretické fyziky MFF UK, jeho zaměstnanci a Jednotou českých matematiků a fyziků.

[24] Lze ho řešit i analyticky ale v jedné dimenzi neobsahuje žádné fyzikálně zajímavé chování, ve třech a více dimenzích není exaktně řešitelný a ve dvou dimenzích existuje exaktní Onsagerovo řešení. K tomu řekněme jen tolik, že Landau údajně kdysi prohlásil: „Celou fyziku jsem si přepočítal od základů sám, jen Osnagerovo řešení Isingova modelu jsem si musel přečíst."

[25] `https://www.matfyz.cz/clanky/1111-aktualita-z-fyziky-uroda-pistacii-podle-isingova-modelu`

[26] Pěkná a přitažlivě graficky zpracovaná ukázka aplikace této teorie je například na webu `http://ncase.me/trust//`.