Úvod do matematického modelování s využitím R

Jiří Kalina, Jiří Hřebíček, Zdeněk Pospíšil, Jaroslav Urbánek

Srpen 2023

Předmluva

Publikace „Úvod do matematického modelování s využitím R“ je vlastně druhým, zásadně aktualizovaným vydáním původního textu „Úvod do matematického modelování s využitím MAPLE“, který vznikl v roce 2010 v souvislosti s řešením projektu ESF č. CZ.1.07/2.2.00/07.0318 „VÍCEOBOROVÁ INOVACE STUDIA MATEMATICKÉ BIOLOGIE“. Ten si kladl za cíl inovovat náplň a provázanost předmětů z kmenového souboru předmětů oboru „Matematická biologie“, studijního programu „Biologie “ Přírodovědecké fakulty MU, garantovaného dvěma výzkumnými institucemi Masarykovy univerzity – Institutem biostatistiky a analýz Lékařské fakulty MU a centrem RECETOX Přírodovědecké fakulty MU.

Oproti koncepci dříve používaného učebního textu: „*Jiří Hřebíček, Michal Škrdla: Úvod do matematického modelování*“ zaměřeného spíše na obecné metody matematického modelování, je předkládaná monografie orientována směrem k praktickému řešení biologických modelů pomocí systému pro statistické výpočty a grafiku R, včetně řešení problematiky neurčitosti a citlivosti parametrů modelu.

Publikace přináší vhodné postupy a metody pro řešení vybraných biologických modelů. Je možné ji využívat ve studiu matematického modelování dvěma způsoby. Studentům, kteří mají větší matematické a informatické znalosti (například v oboru „Matematická biologie“, případně dalších studijních oborů a programů MU) bude pomáhat v tom, jak používat a vytvářet modely v systému R. K tomu jim budou sloužit všechny kapitoly publikace. U studentů s méně pokročilými matematickými a informatickými vědomostmi (typicky studenti dalších biologických či zdravotnických oborů, případně dalších programů MU) nebude třeba studovat kapitolu popisují systém R. Bude jim stačit nastudovat demonstrativní praktické příklady k lepšímu pochopení metodiky matematického modelování.

Vytvořený text pomůže zkvalitnit výuku jednoho z kmenových předmětů oboru „Matematická biologie“ pomocí vybraných speciálních příkladů řešení problémů matematického modelování, kde se využívá nových e-learningových vlastností R. To umožní používat tuto publikaci rovněž vysokoškolským učitelům ke zlepšení jejich pedagogické dovednosti pomocí efektivního využívání systému R při výuce řešení ilustrativních příkladů matematického modelování v biologii.

V Liberci 1. srpna 2023

Jiří Kalina
Zdeněk Pospíšil

ISBN XXX-XXX-XXX-XXX

1 Úvod do matematického modelování a jeho členění

Matematické modelování proniklo do různých oborů přírodních, technických, ekonomických i sociálních věd a stalo se důležitým nástrojem při modelování a simulacích systémů, analýzách a předvídání různých procesů, jevů, chování druhů a stavů společenstev apod. V dalším textu se zaměříme na matematické modelování systémů, kde systémy budeme chápat jako určité abstrakce reálného světa (objektivní reality), které si lidé vytvářejí v procesu jeho poznání. Jako systém budeme zjednodušeně uvažovat (následující výčet je neúplný) např.:

a) *Proces, komplex procesů*, (např. pohyb kyvadla, tok elektrického proudu v obvodu, rozmnožování buněk a organismů, apod.), jímž rozumíme zákonité, na sebe navazující a vnitřně propojené změny nějakého objektu. Proces lze často číselně vyjádřit časovým průběhem nějaké hodnoty, resp. skupin hodnot. Můžeme dávat přednost obecnějšímu vyjádření, kde místo čísel vystupují prvky nějaké množiny. Pak *systémem* nazýváme každý objekt (konkrétní nebo abstraktní), jenž vstupnímu procesu určitého typu přiřazuje výstupní proces téhož nebo jiného typu. Toto přiřazení, které popisuje reakce výstupů na vstupy, se nazývá *chováním systému*, a proto se tato definice nazývá *behavioristická* (behaviour - angl. chování). Místo slova „přiřazení“ často používáme i slovo „transformace“, jež je mnohdy z praktického hlediska výstižnější, protože mnoho systémů opravdu transformuje vstupní procesy na výstupní, tj. přetváří je.

b) Takový *objekt* (přirozený či umělý), který v každém časovém okamžiku má na vstupu nějaký vstupní prvek, na výstupu nějaký výstupní prvek a kromě toho je vždy v nějakém vnitřním stavu, přičemž jsou dány jednoznačné závislosti

   - stávajícího výstupního prvku na stávajícím stavu a vstupním prvku,
   - následujícího stavu na stávajícím stavu a vstupním prvku.

   Tato definice se nazývá *stavová* a je ekvivalentní s první definicí, tj. každý objekt vyhovující definici první vyhovuje i definici druhé a naopak.

c) *Soubor nějakých prvků a vazeb mezi nimi*, např. soužití dravce a jeho kořisti; výroba jeřábu s předepsanou nosností, minimalizace spotřeby vozidla na danou vzdálenost apod. S používáním této definice však nastávají určité potíže. Není snadné interpretovat slovo „vazba“, a ne každý objekt, který lze intuitivně zcela zřejmě považovat za systém, je komponován z několika jasně odlišitelných prvků.

d) *Soubor informačních, regulačních a řídících činností* vztahujících se k a) – c), např.: informační systém, řídící systém, komunikační systém, regulační systém.

e) Abstraktní myšlenkovou konstrukci, výrokovou konstrukci, konstrukci matematických výrazů apod. zaváděném na a) – d).

f) Abstraktní myšlenkovou konstrukci atd. vytvářenou bez přímého vztahu k a) – d).

Použití matematického modelu systému přináší řadu výhod:

- Umožňuje zjistit informace o chování systému, i když ze skutečného systému je to nemožné nebo obtížné.
- Urychluje proces poznání objektivní reality. Procesy, které ve skutečném systému probíhají pozvolna a dlouhodobě, lze pomocí modelu sledovat zrychleně během simulace (výpočtu), která závisí na použité informační a komunikační technologii (ICT).
- Usnadňuje a racionalizuje proces poznání. Matematický model systému dává přehledná, stručná zobrazení objektivní reality a umožňuje postup při řešení problému podle potřeby uživatele. Modely vnášejí nové poznání do našeho myšlení.
- Umožňuje variantní řešení, tj. simulaci a propočet celé řady variant možných výsledků řešení.
- Identifikuje vznik chybného poznání objektivní reality (na rozdíl od experimentu v reálném systému).

Modelování bývá většinou pojímáno jako transdisciplinární činnost, neboť se na něm mohou podílet poznatky z matematiky a fyziky, teorie systémů, teorie pravděpodobnosti, informatiky, kybernetiky či kognitivních věd, operačního výzkumu a jiných. Matematické modelování získává v posledních letech velký význam v praxi, ale také ve výuce studentů na vysokých školách přírodovědného, technického i ekonomického zaměření. Do matematického modelování, stejně jako i do jiných odvětví vědy, proniklo již od šedesátých let minulého století využití informačních a komunikačních technologií (ICT). Nyní si bez jejich využití neumíme matematické modelování představit. Požadavky na tyto počítačové aplikace — symbolické a numerické výpočty, na jejich vysokou přesnost, vizualizaci a interaktivní komunikaci s řešitelem atd. — vedly k vytvoření komplexních aplikačních programů jako jsou např. komerční programy: Matlab od firmy Mathworks Inc.[1], Maple od Maplesoft Inc.[2], MathCAD od MathSoft Inc.[3], Mathematica od Wolfram Reasearch, Inc.[4], MuPAD[5] vyvinutý univerzitou Paderhorne a firmou SciFace Software GmbH a později začleněný do Matlabu atd.

Z volně přístupných (open source) programů zmíňme např. programy: MAXIMA pro symbolické výpočty[6], OCTAVE pro numerické výpočty[7], programovací jazyk Python[8] a R pro statistické výpočty[9]. Tyto aplikační programy lze využít v matematickém modelování během celého vývojového procesu, tj. identifikace, analýzy, vývoje, implementace, řešení a ověřování, případně modifikace matematického modelu.

V současné době se používají některé výše uvedené systémy (např. R, Python, Maple, Matlab a Mathematica) na vysokých školách pro demonstraci probírané látky na cvičeních, případně jsou využívány studenty při individuální přípravě, analýze a řešení problémů, které jim zadávají jejich učitelé nebo vyplývajících z jejich závěrečných prací.

[1] `https://www.mathworks.com`
[2] `https://www.maplesoft.com`
[3] `https://www.mathcad.com`
[4] `https://www.wolfram.com`
[5] `https://www.mathworks.com/discovery/mupad.html`
[6] `https://maxima.sourceforge.net`
[7] `https://octave.org/`
[8] `https://www.python.org`
[9] `https://www.r-project.org/`

Charakteristickým rysem inovace výuky matematického modelování ve studijních programech vysokých škol v České i Slovenské republice, Evropské unii a v zemích OECD se v posledních letech stalo používání nových ICT (např. Grid Computing[10], Cloud Computing[11]) v rámci budování nového vědního oboru „Computational Science“ a „Mathematical Modelling“ [7].

Cílem stále více vysokých škol je ovšem zařadit do výuky předmět seznamující studenty s prací ve výše uvedených systémech, na které mají zakoupenu licenci, a využít jejich možností při návrhu, analýze, řešení a testování netriviálních matematických modelů. To je rovněž cílem této publikace, kde nejprve uvedeme, co je matematický model a metodologie matematického modelování. Na praktických příkladech ukážeme využití systému Maple pro matematické modelování.

Přístup k matematickému modelování s využitím ICT lze rozdělit do „black-box“ modelování[12], „white-box“ modelování a „shadow-box“ modelování, podle toho, v jakém rozsahu jsou předem známy informace o systému a způsobu jeho řešení.

Black-box model je systém, o kterém není známa a priori informace. White-box model je systém, kde jsou známy všechny potřebné informace. Prakticky všechny známé způsoby matematického modelování s využitím ICT náleží mezi black-box a white-box řešení modelů. Nedávno bylo zavedeno tzv. shadow-box modelování, kdy uživatel využívá jako základní black-box modelování a může si zvolit alternativně white-box modelování při řešení modelu systému. Toto umožňuje právě systém Maple, který je v tomto směru nejvíce rozvinut.

Obvykle je lepší používat co nejvíce a priori informací, pokud je to možné, a vytvořit mnohem přesnější matematický model systému. Tudíž white-box modely jsou obvykle považovány za vhodnější, protože pokud se použily informace o systému správně, pak model i jeho řešení se budou chovat správně. Často je apriorní informace dána v podobě znalosti typu funkcí týkající se jejich různých proměnných. Například, když chceme vytvořit model, jak lék funguje v lidském organismu (systému), víme, že obvykle množství léku v krvi v čase je exponenciálně klesající funkce. Víme však, že jsou zde ještě další neznámé parametry určující, jak rychle se množství léku v krvi vstřebá, a jaké je původní množství léku v krvi. Tento příklad tedy není typu white-box model. Jeho parametry musí být odhadnuty prostřednictvím nějakých jiných lékařských metod, a teprve poté je možné použít model s exponenciálně klesající funkcí.

1.1 Matematický model

Matematický model je abstraktní model[13], který využívá matematického jazyka k popisu chování systému. Používá se převážně v přírodních (fyzika, biologie, chemie apod.) a technických (strojírenství, elektrotechnika, stavebnictví apod.) vědách, ale také ve společenských vědách (ekonomie, sociologie, politické vědy apod.). Matematický model transformuje systém do matematického zápisu, který má následující výhody:

- formalizaci zápisu danou historickým vývojem (v současné době je vyvinuta uznávaná mezinárodní standardizace),
- přesná pravidla pro manipulaci s matematickými symboly a strukturami,

[10] `https://en.wikipedia.org/wiki/Grid_computing`

[11] `https://en.wikipedia.org/wiki/Cloud_computing`

[12] `https://en.wikipedia.org/wiki/Black_box`

[13] Abstraktní model (nebo konceptuální model) je teoretická konstrukce, která reprezentuje fyzikální, biologický nebo sociální či ekonomický proces. Sestává z množiny proměnných a množiny logických nebo kvantitativních vztahů mezi proměnnými.

- možnost využití ICT pro zpracování a řešení vytvořeného modelu.

I přes velký potenciál matematického zápisu jím není možné popsat reálné systémy, objekty či procesy, které jsou velmi komplikované. Proto musíme nejdříve identifikovat nejdůležitější části zkoumaného systému, jenž budeme modelovat, a ty musí vytvářený model popisovat. Ostatní prvky systému můžeme buď podstatně zjednodušit, nebo zcela vyloučit.

1.1.1 Základní prvky matematického modelu

Matematický model obvykle popisuje systém s pomocí množiny vstupních a výstupních proměnných, dále parametrů a množiny rovnic, které určují stavy systému a vztahy mezi proměnnými a parametry. Hodnoty proměnných mohou být např. reálná nebo celá čísla, booleovské hodnoty nebo textové řetězce anebo složitější struktury. Proměnné reprezentují nějaké vlastnosti systému, např. výstupy měřených systémů často ve tvaru signálů, vzor- kovaná data, hodnoty počitadla, výskyt dané události či jevu (ano/ne) apod. Na model se můžeme dívat také jako na množinu funkcí, která popisuje vztahy mezi různými proměnnými.

V každém matematickém modelu můžeme rozlišit tři základní skupiny objektů, ze kterých se model skládá. Jsou to:

- proměnné a parametry,
- matematické struktury,
- řešení.

Z hlediska ICT můžeme proměnné a parametry v modelu uvažovat jako:

- **Identifikované (pojmenované) proměnné a parametry.** Identifikovaná proměnná nebo parametr představuje konkrétní vlastnost reálného objektu, pojmenovanou názvem a fyzikální jednotkou, v níž se měří.
  Příklady: x_k je výměra pšenice ozimé v hektarech, x_r produkce pšenice ozimé na parcele „U křížku“ v tunách, c_{ik} vzdálenost dodavatele D_i od spotřebitele S_k v kilometrech.
- **Neidentifikované (pomocné) proměnné a parametry.** Slouží pro formalizaci matematického zápisu, implementaci algoritmů apod. Obvykle se uvažují v bezrozměrných jednotkách.
  Příklad: U lineárního programování se zavádějí pomocné proměnné, které umožní změnit „nerovnice na rovnice“.
- **Nekontrolovatelné proměnné.** Představují procesy v systému, jejichž míry nelze zjistit (jedná se o další typ neurčitosti).
  Příklady: V modelech kilmatu „ad hoc“ jsou charakteristiky počasí nekontrolovatelné parametry nebo proměnné, protože nelze využít počtu pravděpodobnosti pro jejich popis. Velikost míry inflace v chaotických a nestandardních tržních podmínkách nelze popsat ani pomocí pravděpodobnosti ani pomocí fuzzy funkce.

1.2 Klasifikace matematických modelů

Během identifikace a analýzy modelovaného systému je vhodné určit, do jaké kategorie matematický model spadá, což nám umožní snadněji rozpoznat základní vlastnosti a strukturu hledaného modelu.

Podle toho, zda zahrnujeme do modelu náhodné veličiny, lze modely rozdělit do dvou skupin: *deterministických* a *stochastických modelů*. Dále lze tyto skupiny rozdělit dle vztahu k průběhu času (*dynamické*, *statické*) nebo spojitosti (*spojité*, *diskrétní*).

Matematické modely obsahují proměnné, jež jsou abstrakcí hledaných prvků systému a operátorů nad těmito proměnnými. Operátory mohou reprezentovat algebraické operace, funkce, funkcionály, diferenciální operátory atd. Pokud jsou operátory v matematickém modelu *lineární*, hovoříme o *lineárních modelech*, v opačném případě o *nelineárních modelech*.

Můžeme také uvažovat modely se soustředěnými (u homogenních modelů) a distribuovanými (u heterogenních modelů) parametry. Mezi těmito skupinami leží mnoho jiných typů modelů, dále tříděných podle mnoha dalších kriterií, které lze využít.

Matematické modely se používají prakticky ve všech vědách a rozvoj jednotlivých věd je na jejich využívání bezprostředně závislý. Stupeň „matematizace“ vědního oboru je uznávaným měřítkem jeho kvality a zárukou rozvoje. V oblastech přírodních a fyzikálních věd, technice, ekonomii, managementu, marketingu, sociálních a společenských vědách se používá velké množství různých typů matematických modelů, které můžeme klasifikovat podle různých hledisek. Nejobecnější klasifikace dělí matematické modely do dvou skupin:

- **Modely deskriptivní.** Slouží k zobrazení prvků a vztahů v systému a k analýze základních vlastností systému. Nezajímá nás určité cílové chování systému, ale pouze systém sám o sobě. Pomocí těchto typů modelů se odvozují další vlastnosti systému, určuje se jeho rovnovážný stav, stabilní stav, vliv změn uvnitř i ve vnějším okolí systému na jeho chování.
  Příklady: Rovnice $E = mc^2$, soustava diferenciálních rovnic modelující procesy narození a úmrtí, simulační model modelující výskyt škůdců porostu, rovnice nabídky a poptávky v konkurenčním prostředí, ekonometrický meziodvětvový model „Input-Output“ atd.

- **Modely normativní.** Slouží k analýze a řízení systému tak, aby byl splněn nějaký cíl nebo množina cílů. Zajímá nás cílové chování systému. Normativní model bývá často doplněn tzv. cílovou (účelovou) funkcí nebo soustavou takových funkcí. Nutnou součástí normativního modelu je extremální (minimální/maximální) řešení, které dává návod, jak požadovaného cíle (resp. cílů) dosáhnout. Normativní modely, jejichž cílem je nalezení optimálního řešení, se nazývají optimalizační modely.

Modely deskriptivní i normativní jsou dále děleny podle typu systému, k jehož modelování slouží, nebo podle typu matematických složek (proměnné, struktury, řešení), jež obsahují. Tak lze modely dělit podle zohlednění času na:

- **Modely statické.** Model popisuje a analyzuje systém bez zřetele k jeho časovému vývoji. Zobrazení se týká zpravidla určitého časového intervalu (týden, měsíc, rok apod.).

- **Modely dynamické.** Model popisuje a analyzuje systém v průběhu času. Zobrazení může být typu „ex post“ nebo „ex ante“ a respektovat krátký či delší časový horizont.

- **Modely dynamizované.** Zpravidla se jedná o zahrnutí faktoru času do dosud statického modelu pomocí speciálních modelových technik. Dynamizované modely se používají v případě, kdy odpovídající dynamický model je velmi složitý nebo jej nedovedeme soudobými modelovými technikami spolehlivě konstruovat. Příklad: U lineárního programování k pravé straně maticové rovnice $A \cdot x \leq b$ přidáme časový vektor $u(t)$.

Nebo podle metod, kterými přistupují k náhodným dějům a nejistotě na:

- **Modely deterministické.** Všechny proměnné, parametry a funkce v modelu jsou deterministické (nenáhodné) veličiny nebo funkce.

- **Modely stochastické.** Alespoň jedna proměnná, parametr nebo funkce v modelu je náhodná veličina nebo náhodná funkce.

- **Fuzzy modely.** Některé proměnné a parametry jsou „fuzzy“ (nepřesně ohraničené, neostré, neurčité, vágní). Funkce příslušnosti ve fuzzy logice jim umožňuje přiřadit příslušnost k množinám v rozmezí od 0 do 1, včetně obou hraničních hodnot. Uplatnění fuzzy modelování je účelné ve všech případech, kdy se řeší problém spojený s neurčitostí, s nepřesností a musí se pracovat s neurčitými daty a používání přesných popisů by vedlo k idealizování skutečností reálného světa a tedy k odklonu od reality.

Podle povahy problému se modely používají individuálně nebo v kombinacích. Pro řešení známých problémů lze použít tzv. standardní modely. Pro řešení nových problémů je třeba konstruovat nové modely.

1.3 Modelování neurčitosti, nejistoty a rizika

1.3.1 Modelování neurčitosti

Neurčitost zde chápeme jako vlastnost systému, kdy jej nelze přesně matematicky popsat nebo kdy matematický popis neumožňuje dostatečně přesně predikovat jeho budoucí chování. Obecně se dá říci, že zdrojem neurčitosti je nedostatek informací. Problémem je, že informace, ze kterých při tvorbě modelu vycházíme mohou být nekompletní, vzájemně si odporující, nespolehlivé, vágní nebo jinak nedostačující. Tyto nedostatky v potřebných informacích mají za následek různé druhy neurčitostí. Nejistotou při transformaci systému do matematického modelu rozumíme situaci, kdy nemáme k dispozici všechny potřebné informace nebo kdy některé z informací jsou nespolehlivé.

Pro jednoduchost můžeme neurčitosti ovlivňující model rozdělit na tři spolu související kategorie [33]:

a) **neurčitost v matematickém popisu modelu** je výsledkem nedostatečné znalosti chování modelovaného systému, neúplných exaktních dat, či zjednodušení, které bylo nutné provést pro matematický popis. V literatuře je možné setkat se s pojmy strukturální (konstrukční) chyba, konceptuální chyba, neurčitost v konceptuálním modelu, nebo chyba/neurčitost modelu, které částečně či zcela odpovídají tomuto druhu neurčitosti.

b) **datová neurčitost** neboli neurčitost v datech je způsobena chybami měření, nepřesností analytických metod a omezeným množstvím vzorků při sběru a zacházením s daty. Datovou neurčitost někdy nazýváme odstranitelnou neurčitostí, neboť je možné ji dalším studiem (měřením) minimalizovat. Speciálně pro tento druh neurčitosti používáme v češtině termín nejistota.

c) **neurčitost v aplikaci modelu** vyjadřuje neurčitost (chybu), která vznikne použitím modelu. V tomto případě se jedná zejména o řešení matematických rovnic popisujících model pomocí nástrojů ICT.

Analýza neurčitostí vyšetřuje zmíněné neurčitosti ve vstupu a jejich vliv na výstup modelu. Zpravidla se skládá z následujících kroků:

- *charakterizace vstupních neurčitostí* – odhad neurčitostí ve vstupu a parametrech algoritmu modelu;
- *šíření neurčitostí* – odhad neurčitosti ve výstupu způsobené neurčitostmi na vstupu modelu;
- *charakterizace neurčitostí modelu* – charakterizace neurčitostí spojená s jinými strukturami algoritmu modelu a formulacemi modelu;
- *charakterizace neurčitostí v predikcích algoritmu modelu* – vychází z neurčitostí ve vyhodnocených datech.

Neurčitost má (nejméně) dvě vzájemně komplementární stránky: *vágnost* a *nejistotu*. Vágnost lze modelovat např. pomocí teorie *fuzzy množin*, zatímco nejistotu např. pomocí *teorie pravděpodobnosti* a popř. dalších teorií, jako je teorie možnosti, různé míry věrohodnosti apod. Můžeme tedy říci, že pravděpodobnost nám odpovídá na otázku, zda „něco nastane", zatímco teorie fuzzy množin nám odpovídá na otázku, „co vlastně nastalo". Je zřejmé, že při matematickém modelování systému s neurčitostmi se vyskytuje jak nejistota, tak vágnost. Ze studijních účelů však lze obě tyto stránky oddělit a zabývat se pouze jednou z nich.

Podle [25] můžeme říci, že „*Předmětem teorie pravděpodobnosti je studium a modelování nejistoty. Ta nastává tehdy, jestliže se setkáváme s nějakým jevem, který může, avšak nemusí nastat. Nemáme tedy jistotu, že jev opravdu nastane. Základním pojmem v teorii pravděpodobnosti je rozdělení pravděpodobnosti. To charakterizuje způsob nastání jevů vybíraných z nějaké množiny různých jevů, o nichž víme určitě jen to, že jeden z nich nastane. Pravděpodobnost nám pak dává informaci o tom, zda nastání některého z uvažovaných jevů můžeme očekávat s větší jistotou, než nastání jiného jevu.*"

Naproti tomu uvažujme např. *objekty s určitou vlastností* a na otázku, jakou mají „*vlastnost*" odpovíme, že by ji mohly mít, než, že ji mají či ne. Jde totiž o vymezení vlastnosti a nikoliv toho, zda ji jev má či ne. Základním pojmem je zde fuzzy množina objektů a funkce příslušnosti objektu do ní. To znamená, že prvek patří do množiny s jistou mírou příslušnosti – stupněm příslušnosti. Funkce, která každému prvku universa přiřadí stupeň příslušnosti, se nazývá funkce příslušnosti. Funkce příslušnosti, stejně jako pravděpodobnosti, mohou nabývat hodnot z intervalu $[0, 1]$. To je však jen vnějšková shoda s teorií pravděpodobnosti.

„Fuzzy logika umožňuje zahrnout nepřesnost a poměrně jednoduchým způsobem pracovat s významy slov přirozeného jazyka. Používá vágně charakterizované expertní znalosti. Tedy pravý opak toho, co se vždy požadovalo – větší přesnost. Narážíme na reálný rozpor, jehož řešení neexistuje. Jde o vztah mezi relevancí a přesností informace. Princip, který L. A. Zadeh nazval principem „inkompatibility", lze charakterizovat takto: Chceme-li popsat realitu, pak musíme rozhodnout mezi relevancí informace, která bude méně přesná, nebo přesností informace, která však bude méně relevantní. Při zvyšování přesnosti se dostaneme k bodu, kdy přesnost a relevance se stávají vzájemně se vylučujícími charakteristikami." [25]

Příklad: Instrukce k zaparkování auta: „pootoč kola o 19° 25,32' a popojeď o 368,1256 mm dozadu" – jednak by se tato informace zdlouhavě a složitě chystala a také by bylo obtížné ji přesně splnit. Stačí říct: „pootoč kola mírně doleva a popojeď o malý kousek dozadu." „*K vyjádření relevantní informace je nezbytné použít přirozený jazyk. Je to dosud jediný dokonalý prostředek, který nám umožňuje efektivně pracovat s vágními pojmy.*" [25]

Ukazuje se, že přesnost matematického modelu je pouze iluze, neboť je principiálně nedosažitelná. Snaha o absolutní přesnost nás vždy dovede ke sporu. Není však třeba litovat, neboť vágnost, kterou přirozený jazyk umí dokonale využít, je jeho hlavní silou, nikoliv nedostatkem.

Častá námitka, že to, co je řešeno pomocí fuzzy logiky, lze řešit i bez ní, neobstojí. Rozdíl mezi klasickým řešením a řešením pomocí fuzzy logiky je v čase a s tím souvisejících

nákladech. Trvalo by to mnohem déle, abychom dosáhli stejného efektu (správnosti). Nalezení matematického popisu může být v praxi velmi obtížné. Často je popis velmi složitý, a následné použití modelu není zrovna snadné. Proto se buď používají přibližné metody nebo se přijímají různá zjednodušení a výsledek pak nemusí být uspokojivý.

Při řešení pomocí fuzzy logiky je navíc větší jistota, že dané řešení (model systému) bude robustnější vzhledem k náhodným poruchám a nepředvídaným situacím, které pochopitelně lze očekávat.

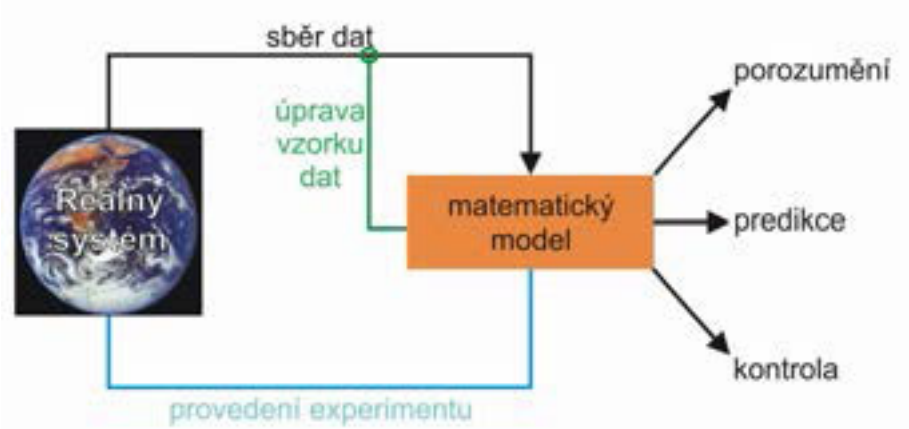
K vyjádření relevantní informace je možné použít přirozený jazyk. S počítačem však není možné komunikovat v přirozeném jazyce, a proto jsou nezbytná jistá zjednodušení. Obvyklý způsob je použití pravidel typu JESTLIŽE-PAK. Tato pravidla jsou základem všech úvah a základem činnosti všech algoritmů.

Příklad: JESTLIŽE sklon svahu je velký a srážky jsou intenzivní, PAK se svah velmi rychle sesune.

Modelování při riziku předpokládá, že některé informace jsou náhodné veličiny, nebo že některé procesy jsou popsány náhodnými funkcemi. V případě modelů s rizikem můžeme velikost rizika při přijetí řešení popsat pomocí pravděpodobnostních charakteristik [25].

2 Metodologie matematického modelování

Na obrázku 2.1 je znázorněn proces zkoumání systému, který začíná monitoringem a sběrem dat o systému, pokračuje jejich vstupem do matematického modelu a na základě analýzy výsledků řešení modelu (porozumění, predikce a kontrola jeho chování), je provedena případná úprava modelu a přizpůsobení sběru dat. Pokud to nevede k uspokojivému řešení, provedou se na zkoumaném systému nové experimenty, případně se modifikuje matematický model.



Obrázek 2.1: Proces zkoumání systému s využitím matematického modelování

2.1 Obecné zásady matematického modelování

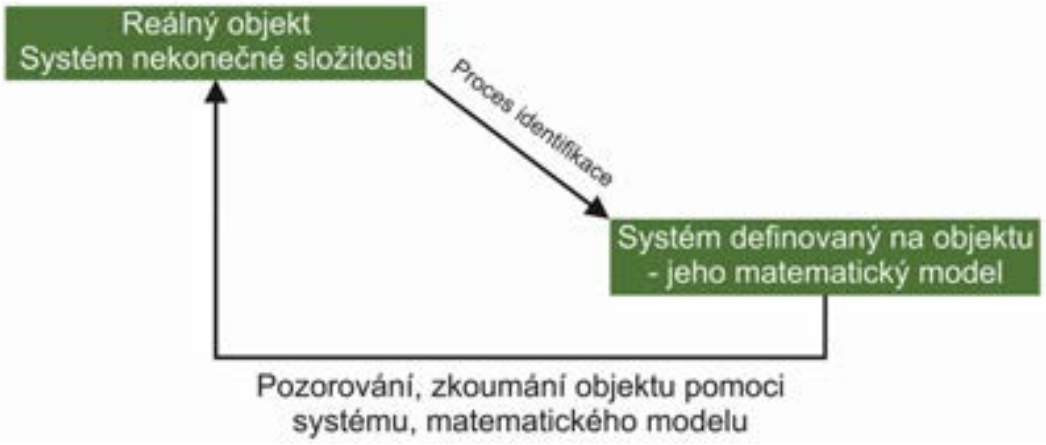
Matematické modelování je odborná a kvalifikovaná činnost vyžadující týmovou spolupráci odborníků z různých oblastí: odborníka z oblasti oboru řešené problematiky, specialistu v oblasti matematiky, specialistu z oblasti informatiky apod. Pro úspěšné matematické modelování musí být splněny následující předpoklady:

- Znalost metod a prostředků matematické a funkcionální analýzy, algebry a diskrétní matematiky — důležité pro volbu správné metody řešení a modelu.
- Znalost techniky modelování a zdrojů informací o modelu. Úsilí vynaložené na konstrukci a využití určitého modelu z literatury musí být úměrné jeho přínosu.
- Týmová spolupráce. Musí existovat dostatečný prostor pro vlastní vývoj matematického modelu (iniciativa) a musí být zainteresovanost (studijní, výzkumná) na využití modelové techniky (motivace).

- Výpočetní základna. Všechny tři složky ICT (tj. hardware, software a komunikace) musí být řešitelskému týmu k dispozici a musí být v rovnováze.
- Informační a datová základna. Každý model je třeba ověřit pomocí vstupních proměnných, parametrů a dat, které vycházejí z konkrétních hodnověrných informací, dat a zdůvodněných odhadů parametrů. Údaje musí být ve vhodné formě pro ověřování modelu. Je vhodné vytvářet speciální informační systémy (např. banky dat), jež uchovávají data.
- Experimentální základna. U složitých matematickým modelů by měla být k dispozici i experimentální základna, kde se řešení modelu ověří v praxi.

Metodologie matematického modelování se vyvíjí jako samostatná odborná specializace, která se v současné době zařazuje do studijních programů Matematického a počítačového modelování na vysokých školách. Systém uvažujeme jako uspořádanou množiny prvků, mezi nimiž působí vzájemné vazby. Obecné zásady, jež je třeba při matematickém modelování systémů respektovat, lze velmi zjednodušeně popsat následujícími kroky, které pak podrobněji popíšeme v další kapitole:

1. Identifikace systému z hlediska matematického modelování sestává z:
   - Rozhodnutí, zda se jedná o standardní systém (problém), již řešený a volba standardního modelu.
   - Rozhodnutí, zda se jedná o nový, dosud neznámý systém, a zda použijeme upravený standardní model nebo vytvoříme model nový. K tomu je třeba zpravidla vytvořit tvůrčí odborný tým.
   - Rozhodnutí, zda model bude statický, dynamický, dynamizovaný, deterministický, stochastický. Zda bude deskriptivní, nebo normativní. Zda systém bude modelován jedním modelem či více modely a jak budou vzájemně uspořádány (propojeny).



Obrázek 2.2: Identifikace systému

   Identifikace systému je pojem, který se obvykle používá k popisu matematických nástrojů a algoritmů, jež vytváří dynamické modely z naměřených dat. V této publikaci jej budeme uvažovat v obecnějším smyslu.

2. V kroku specifikace modelu systému je nutno určit:
   - Prvky systému: tj. elementární část systému při dané rozlišovací úrovni dále nedělitelná.

- Vazby: vzájemné závislosti mezi prvky systému, tj. kauzální vztahy: příčina-následek, způsoby spojení mezi prvky, souvislosti mezi jevy, informační vazby, matematicky formulované vztahy atd.
- Strukturu systému: je dána prvky a vazbami mezi nimi.
- Stavy systému: tj. popsat stavy chování systému a přechodu mezi nimi u dynamických systémů apod.
- Okolí systému: tj. množinu prvků, které nejsou prvky systému, ale mají k němu významné vazby.
- Organizaci dat v systému: tj. jejich strukturu, způsobu uložení, vstup a výstup dat atd.
- Verifikaci modelu systému: tj. ověření matematických předpokladů, stability, konzistence a konvergence řešení atd.

Konkrétní specifikace či formulace matematického modelu je základem celého matematického modelování a záleží do značné míry na schopnostech řešitelského týmu spojit teoretické poznatky s informacemi o konkrétním problému nebo systému, který je předmětem analýzy. V této fázi matematického modelování musí být věnována pozornost i tomu, zda vstupní data skutečně odpovídají proměnným zahrnutým do modelu. V některých případech je totiž vhodnější dát přednost relativně jednoduše specifikovanému modelu před složitým, často zavádějícím modelem.

3. Výpočet řešení modelu sestává z:
   - Volby algoritmu řešení.
   - Výběru variant řešení.
4. Výběr dostatečně vhodných řešení sestává z:
   - Výběru vhodných řešení v rámci algoritmu řešení.
   - Výběru vhodných řešení prováděných specialistou v řešené problematice.
   - Výběru vhodných řešení prováděných skupinou expertů.
5. Experimentování s vybraným řešením sestává z:
   - „What-if“ analýzy, tj. analýzy „Co se stane, když“ používané většinou při numerické simulaci sloužící k odhalení, jak je model citlivý na odhad vstupních parametrů, jež by měly ležet v nějakém vhodném intervalu.
   - „Goal seeking“ problému, tj. „Cílovým hledáním problému“, který souvisí s tím, že v praxi se vytváří matematický model metodou postupných kroků lépe aproximujících řešený systém. Toho se dosahuje lokálním hledáním dostatečně přesného řešení. Tento problém je v anglické literatuře nazýván jako „satisfying problem“, „feasibility problem“, nebo také „goal seeking“ problém.
   - Scénářů, tj. vhodnou volbou parametrů modelu se zkoumají různá řešení (scénáře), z nich se pak vybere nejvhodnější řešení.
6. Výběr optimálního řešení, tj. na základě předem stanovených kritérií se vybere optimální řešení.
7. Implementace, tj. model je nutno implementovat ve vhodném softwarovém prostředí (vývojové prostředí, programovací jazyk, vizualizace řešení apod.). Přitom je nutno:

- Sledovat postup implementace modelu.
- Připravit experimentální data s cílem ověřit implementovaný model a provést jejich validaci.
- Analyzovat počítačové výstupy řešení modelu a mít zpětnou vazbu na parametry modelu při volbě scénářů.
- Na základě získaných výsledků provést úpravy modelu a jeho novou implementaci.

2.2 Matematické modelování s využitím ICT

Postup při matematickém modelování reálného problému s ICT je uveden na obrázku 2.3 a sestává z několika kroků. Jde o permanentní interaktivní proces s četnými zpětnými vazbami, který se několikrát opakuje [11]. Nejprve je nutno vyjasnit si cíle, které chceme dosáhnout. Ty určí směr řešení ve dvou bodech:

- Na jaké úrovni podrobnosti chceme zkoumat objekt.
- Za druhé musíme vytvořit hranici mezi modelovaným systémem a jeho přirozeným prostředím. Toto rozdělení je správné, pokud prostředí ovlivňuje chování systému, ale modelovaný systém neovlivňuje toto prostředí.

2.2.1 Identifikace modelu

Identifikace (stanovení) jednotlivých prvků matematického modelu s využitím odborné literatury (knihy, časopisy, Internet), spoluprací s odbornou a vědeckou komunitou a dále s využitím ICT k vyhledání a sdílení znalostí o řešeném problému je prvním krokem, který musíme udělat při vytváření matematického modelu. Správná matematická formulace zkoumaného problému je velmi důležitá pro další postup řešení. Je třeba vyjít z analýzy systému, z jeho celkového chování a stanovených cílů řešení. Realita je složitá, je třeba ji vymezit a pro účely modelu zjednodušit. Proto definujeme v rámci objektivní reality systém, tj. prvky, vazby, vstupy a výstupy, procesy, stavy a funkce. Dále provádíme zjednodušení (simplifikaci) řešeného problému, kdy nepodstatné oddělujeme od podstatného.

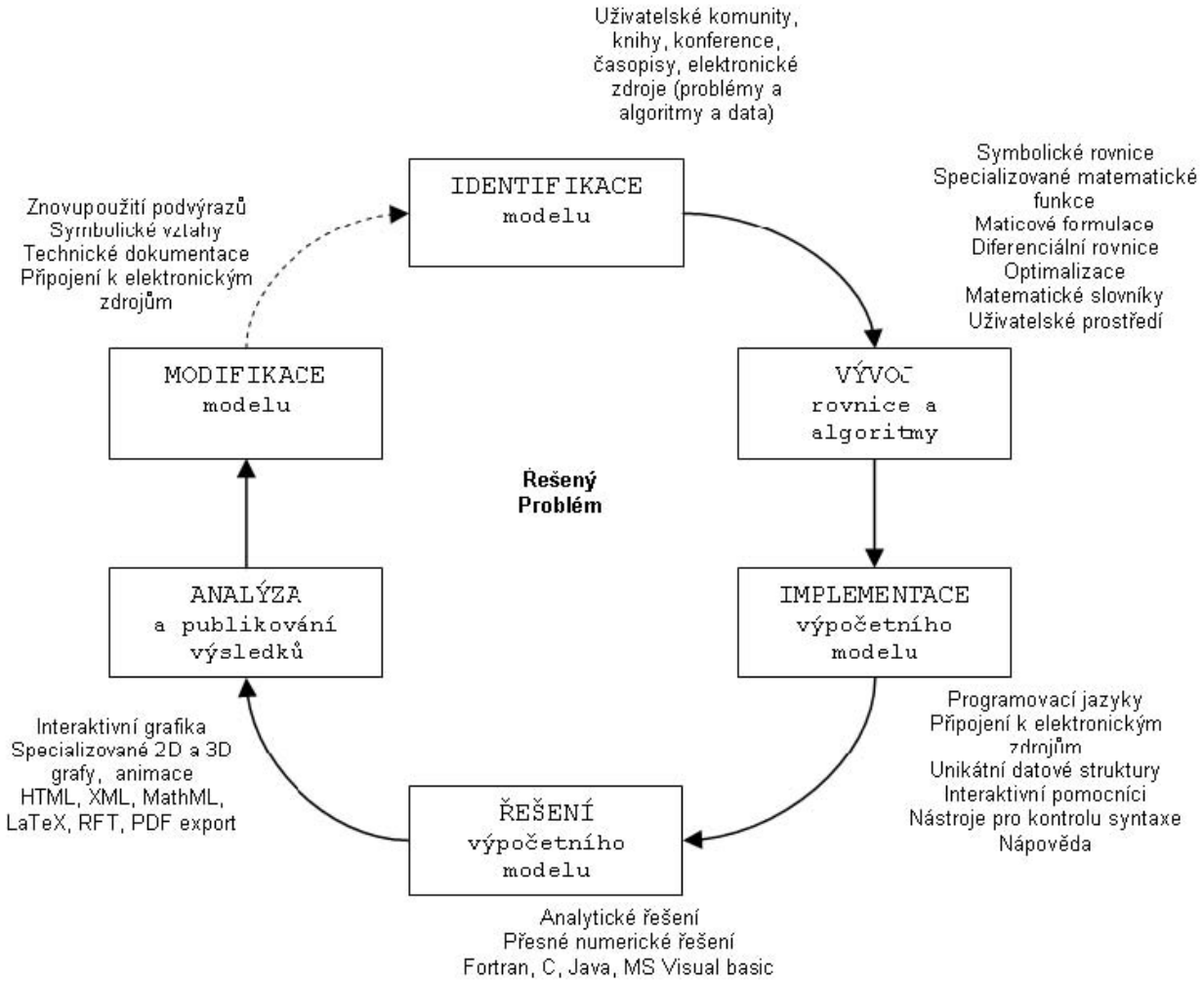
Tvorba předpokladů

Když jsme rozhodli o modelovaném systému, musíme vytvořit základní strukturu modelu, která musí reflektovat naše předpoklady a domněnky o tom, jak systém funguje. Tyto domněnky mohou být uvedeny do formy základních předpokladů. Budoucí analýzy systému vždy zachází s těmito předpoklady jako s pravdivými, ale výsledky těchto analýz budou validní, pouze pokud tyto předpoklady jsou platné.

Newton předpokládal, že hmotnost je obecně konstantní, kdežto Einstein uvažoval hmotnost jako proměnlivou. To je jeden z elementárních rozdílů mezi klasickou mechanikou a teorií relativity. Aplikací výsledků klasické mechaniky na objekt pohybující se rychlostí blízkou rychlosti světla vede k nesrovnalostem mezi teorií a pozorováním. Pokud jsou předpoklady dostatečně precizní, mohou okamžitě vést přímo k matematickým rovnicím popisujícím modelovaný systém.

2.2.2 Sestavení modelu

Sestavení modelu (vývoj matematických rovnic a formulí) včetně matematické analýzy (korektnost, konzistence, stabilita a konvergence řešení) je dalším důležitým krokem v matematickém modelování. Pro konstrukci modelu je rozhodující účel, který sledujeme. Ten



Obrázek 2.3: Matematické modelování s využitím ICT

rozhoduje o tom, co budeme ve skutečnosti pokládat za významné (co zahrneme do modelu) a co jako podružné ponecháme mimo model a mimo naše úvahy. Důležitá je zde analýza citlivosti jednotlivých parametrů modelu a minimalizace jeho neurčitosti. Tvorba modelů patří k tvůrčí činnosti a vyjadřuje kromě dobré znalosti modelové techniky také dobrou znalost věcné problematiky. Každý model musí vycházet z konkrétní hypotézy odvozené z objektivní reality (skutečnosti).

Výběr matematických rovnic

Poté, co bylo rozhodnuto o struktuře modelu, musí být vybrány matematické rovnice k popsání modelovaného systému. Je velmi rozumné vybrat tyto rovnice pečlivě, protože mohou nepředvídatelně ovlivnit chování matematického modelu.

Matematické rovnice z odborné literatury a Internetu

Může se stát, že někdo další publikoval matematické rovnice týkající se problému, o nějž se zajímáme. To poskytuje dobrý výchozí bod, ale je nezbytné postupovat s těmito informacemi obezřetně. Problémy, s nimiž se můžeme setkat, jsou následující:

- Rovnice jsou odvozené z dat v oblasti vysvětlujících proměnných, která neobsahuje oblast nutnou pro aplikaci modelu.

- Experimentální podmínky (prostředí) se podstatně liší od podmínek vyskytujících se v našem modelovaném problému.

- Rovnice popisují chování většiny dat, aniž by uvažovaly známé odchylky na koncích oblasti dat, nebo jejich variabilitu (proměnlivost).

Některé oblasti vědy jsou dostatečně dobře prostudované, a tak odpovídající formulace analýz se staly standardem. Proto je relativně bezpečné uvažovat podobné analýzy (a proto i struktury rovnic) za řešení podobných problémů.

Často nejsou rovnice v literatuře vyjádřeny v přesné formulaci pro hledaný model. Pojmenované a pomocné proměnné mohou být přesunuty během regrese. Rovnice také může popisovat změnu váhy zvířat vzhledem k času, přestože model potřebuje znát změnu počtu jedinců v čase. V jiném případě můžeme přijmout parametr pouze odhadující přibližnou hodnotu, protože není nejdůležitějším pro naše potřeby.

2.2.3 Implementace modelu

Implementace modelu s využitím ICT (naprogramování v příslušném programovacím jazyce, jeho odladění a verifikace, analýza jeho výpočetní složitosti, využití příslušného hardware atd.). Zde se vyplatí využít moderních ladících technik, případně i použít již vyvinuté a dostupné (open source) knihovny, služby i moduly z Internetu.

2.2.4 Řešení modelu

Řešení implementovaného modelu s využitím ICT nástrojů (analytické, numerické atd.). Naplnění modelu konkrétními parametry a daty. Je třeba dbát na jejich hodnověrnost. Existují dva způsoby odvození řešení z modelu:

a) Analytické (explicitní) řešení spočívá v nalezení přesného řešení pomocí analytických matematických metod (řešení soustav rovnic, řešení úlohy na vázaný extrém apod.).

b) Numerické (přibližné) řešení se používá při řešení modelů, u kterých neumíme problém řešit analyticky, nebo v případech, kdy je analytické řešení obtížné a složité (metody Monte Carlo, simulace na počítači apod.). Při numerickém řešení musíme uvažovat jeho numerickou stabilitu, konvergenci a chybu, která nám vznikne.

2.2.5 Analýza řešení modelu

Verifikace řešení (kontrola, zda výsledky souhlasí s chováním objektu), jeho vizualizace atd. a publikace výsledků. Model je jen přibližným obrazem objektivní reality. Je dobrý, jestliže umožní přesně sledovat důsledky změn ve vstupech do systému na výslednou efektivnost systému. Cílem testování modelu je prověření jeho správné struktury, vypovídací schopnosti, formálních kvantitativních vlastností včetně odstranění formálních chyb. Testování modelu provádíme tak, že modely naplníme empirickými číselnými údaji, dosažené výsledky analyzujeme a porovnáváme s realitou. Ověřování lze promítat i do minulosti („ex post“) i do budoucnosti („ex ante“). Interpretační analýza představuje převod výsledků do reálného systému. Je to aktivní proces, při kterém je třeba provádět neustále logickou kontrolu smyslu řešení, vyhnout se nebezpečí mechanického používání modelové techniky. Významným prvkem interpretace je promítnutí výchozích hypotéz a předpokladů do výsledku řešení. Shrnutí získaných poznatků včetně všech aspektů, které nebyly do matematického modelu zahrnuty.

2.2.6 Modifikace modelu

Modifikace modelu a jeho vylepšení. V případě, že dosažené řešení není v dostatečném souladu s objektivní realitou, je nutno začít znovu postupovat od kroku 1 a opakovat celý předešlý postup matematického modelování do té doby, dokud nedosáhneme uspokojivého řešení zkoumaného problému.

Metody prezentace modelu jeho potenciálním uživatelům závisí na znalostech uživatele modelu a matematického modelování obecně. Pokud chce uživatel vědět raději méně o detailech modelu, je vhodné ukázat mu všechny relevantní informace o výstupech modelu. To umožní uživateli (který není programátorem) vytvořit si objektivnější pohled na řešení modelu a jeho interpretaci. Je dobré zkontrolovat, zda predikce řešení zahrnují skryté extrapolace. Tyto extrapolace mohou hrát úlohu buď vzhledem k použitým datům při vytváření modelu, nebo vzhledem k datům použitým při testování modelu.

3 Populační modely

Teorii matematického modelování popsanou v úvodních kapitolách nyní aplikujeme na konkrétním případě tzv. populačních modelů. Budeme vytvářet modely populací živých organizmů, které žijí v daném čase a prostředí. Naším cílem bude vytvořit model odpovídající na otázku, kolik jedinců bude mít populace v daném čase $t > 0$, jestliže známe tento počet na počátku (v čase $t = 0$).

Modely růstu populace patří k nejrozšířenějším a nejznámějším, tudíž bychom mohli nyní převzít nějaký již vytvořený model z odborné literatury (např. [15]) nebo ze zdrojů na Internetu (např. [19], [28]). Předpokládejme však, že žádný takový model ještě vytvořen nebyl a popíšeme podrobně jednotlivé kroky jeho tvorby.

3.1 Růst populace živých organismů

3.1.1 Identifikace a sestavení modelu

Nejprve budeme modelovat růst jedné populace. Cíl modelu jsme již stanovili. Nyní musíme specifikovat jednotlivé prvky modelu a definovat předpoklady, za nichž bude model platit. Modelujeme růst jedné populace P v čase t, která sestává z počtu N jedinců a na počátku (v čase $t = 0$) měla N_0 jedinců. Předpokládejme, že změna velikosti N populace P v čase je způsobena pouze plozením nových jedinců a umíráním jiných. Přitom počet nově narozených, respektive zemřelých jedinců bude přímo úměrný velikosti populace. Pro jednoduchost budeme uvažovat, že na populaci P nepůsobí žádné další vlivy.

Hledejme řešení modelu, tj. velikost N populace P v daném čase t. Čas t budeme uvažovat jednak jako diskrétní veličinu nabývající celočíselných hodnot (mohou představovat například roky), nebo jako spojitou veličinu.

Na základě vyslovených předpokladů jsme schopni sestavit rovnici modelu. Označme:

$N(t)$ ... funkci představující počet jednotlivců populace (velikost populace) v čase t
a ... *koeficient porodnosti* populace
(podíl nově narozených jedinců ke všem jedincům za jednotku času)
b ... *koeficient úmrtnosti* populace
(podíl zemřelých jedinců ke všem jedincům za jednotku času)
h ... kladné reálné číslo představující časový interval

Vzhledem k smysluplnosti modelu předpokládáme, že všechny výše uvedené proměnné mohou nabývat pouze reálných nezáporných hodnot, navíc $b \leq 1$ (nemůže zemřít více jedinců, než kolik jich je v populaci). Počet jedinců $N(t)$ vypočítaný pro daný čas t nemusí být celé číslo, při interpretaci výsledků jej proto budeme zaokrouhlovat. Navíc předpokládáme, že modelujeme růst populace od času $t = 0$, v němž známe hodnotu velikosti populace $N(0) = N_0$.

Počet nově narozených jedinců za časový interval $[t, t+h]$ je přímo úměrný počtu jedinců $N(t)$ populace v čase t a délce h tohoto intervalu, tj:

$$a \cdot N(t) \cdot h,$$

kde koeficientem přímé úměrnosti a je koeficient porodnosti. Analogické tvrzení platí pro počet zemřelých jedinců za daný časový interval, tj:

$$b \cdot N(t) \cdot h,$$

kde koeficientem přímé úměrnosti b je koeficient úmrtnosti.

Nyní již můžeme formulovat rovnice modelu.

Diskrétní případ

Čas t v tomto případě nabývá pouze celočíselných hodnot, takže h bude rovněž celé kladné číslo. Pro jednoduchost zvolme $h = 1$. Rovnice modelu má pak tvar[14]:

$$N(t+1) = N(t) + a \cdot N(t) \cdot 1 - bN(t) \cdot 1 = (1 + a - b) \cdot N(t). \tag{3.1}$$

s počáteční podmínkou

$$N(0) = N_0. \tag{3.2}$$

Spojitý případ

Nechť h je nyní kladné reálné číslo. Potom má rovnice modelu tvar:

$$N(t+h) = N(t) + a \cdot N(t) \cdot h - b \cdot N(t) \cdot h. \tag{3.3}$$

Jelikož uvažujeme spojitý čas, tak můžeme předpokládat, že časový interval $[t, t+h]$ je nekonečně malý. Proto upravíme rovnici (3.3) na tvar:

$$\frac{N(t+h) - N(t))}{h} = a \cdot N(t) - b \cdot N(t) = (a - b) \cdot N(t).$$

A odtud limitním přechodem $(h \to 0)$ dostaneme:

$$\lim_{h \to 0} \frac{N(t+h) - N(t))}{h} = (a - b) \cdot N(t). \tag{3.4}$$

Výraz na levé straně rovnice 3.4 je derivace funkce $N(t)$ podle proměnné t, takže výsledná rovnice modelu má tvar[15],[16]:

[14] Tomuto typu rovnic říkáme rovnice rekurentní (případně diferenční) [17].

[15] Tomuto typu rovnic říkáme rovnice diferenciální [16].

[16] Derivaci funkce podle času budeme dále značit apostrofem, tj. $\frac{d}{dt}N(t) = N'(t)$.

$$\frac{d}{dt}N(t) = (a - b) \cdot N(t) \tag{3.5}$$

s počáteční podmínkou

$$N(0) = N_0. \tag{3.6}$$

Z biologického hlediska lze limitně přejít k diferenciální rovnici 3.5 pouze, pokud jsou splněny následující podmínky [9]:

- kvantovací podmínka: populace je tak velká, že není třeba počítat s jedinci,
- vzorkovací podmínka: všichni jedinci v populaci jsou identičtí (populace je homogenní z hlediska jedinců v produkčním věku, např. neuvažujeme jejich pohlaví).

3.1.2 Implementace modelu a jeho symbolické řešení

Řešení výše uvedeného matematického modelu (rovnic modelu) dokážeme vypočítat sami na základě svých znalostí, užitečný by nám ovšem mohl být také některý z programů vhodných pro symbolické výpočty (tedy výpočty, kde proměnné nejsou v paměti počítače reprezentovány konkrétními čísly). V úvodu (kapitola 1) jsme zmínili tzv. CAS (Computer Algebra System) systémy, např. Maple. MAXIMA nebo MuPAD. Prostředí jazyka R není pro provádění symbolických výpočtů určeno, přesto v něm můžeme pomocí následujícího „triku“ demonstrovat řešení našeho modelu a ověřit si tak jeho správnost. Využijeme k tomu již dříve též zmíněný programovací jazyk Python a pomocí knihovny `reticulate` [34] si z něj, resp. z jeho balíku `sympy` [24] „vypůjčíme“ funkce `rsolve()` resp. `dsolve()` pro symbolické řešení diferenčních, resp. diferenciálních rovnic. Pro spuštění uvedeného příkladu je nicméně zapotřebí nainstalovat na počítač prostředí jazyka Python, což překračuje rámec našeho textu, a proto ponecháváme příklad jen jako demonstrativní ukázku, která nijak nepodmiňuje pochopení dalšího textu. V následujících příkladech se mimo jedné výjimky k Pythonu nebudeme vracet a místo symbolických řešení se spokojíme s řešeními numerickými, která jsou pro prostředí R přirozená. Konkrétně budeme používat výpočetní prostředí jazyka R[17] (více o práci v systému v kapitole REF R).

Před samotným symbolickým řešením modelu pomocí Pythonu bude v prostředí R nutné provést instalaci a načtení knihovny `reticulate`:

```
install.packages("reticulate")
library(reticulate))
```

a po vybudování spojení mezi R a Pythonem dále nainstalovat a načíst pythonovskou knihovnu `sympy` a naimportovat v ní obsažené příkazy a funkce (upozorňujeme, že následující kód je směsí jazyka R vně a jazyka Python uvnitř uvozovek):

```
py_install("sympy")
import("sympy")
py_run_string("from sympy import *")
```

takto zpřístupněné funkce a příkazy knihovny `sympy` nyní využijeme za pomoci příkazu `py_run_string()`, který umožňuje z prostředí R spustit libovolný řetězec kódu v Pythonu, k vyřešení diskrétní i spojité varianty modelu:

[17] Konkrétně se jedná o verzi R-4.2.3 a uživatelské rozhraní RStudio verze 2023.03.0. Vzhledem k tomu, že veškerá práce v prostředí R a software RStudio probíhá v angličtině, upozorňujeme na rozdílný oddělovač desetinných míst v textu (čárka v češtině) a ve zdrojovém kódu R (tečka v angličtině).

Diskrétní případ

V rovnici modelu 3.1 vystupují proměnné a, b a čas t a časově závislá funkce $N(t)$, které nadefinujeme v prostředí Pythonu a následně definujeme funkci f jako levou stranu upravené rovnice 3.1, na jejíž pravé straně je 0. Pak už nám nic nebrání zavolat příkaz `rsolve()`, který nám vyřeší rekurentní rovnici:

```
py_run_string("from sympy.abc import a,b,t")
py_run_string("N=Function('N')")
py_run_string("f=N(t)-(1+a-b)*N(t-1)")
py_run_string("print(rsolve(f,N(t)))")
```

získáváme tak řešení ve tvaru `C0*(a - b + 1)**t`, které snadno interpretujeme (operátor `**` se v Pythonu používá pro umocňování). Zbývá nám vysvětlit konstantu `C0`, která je součástí obecného řešení, a která v našem případě odpovídá počáteční velikosti populace N_0 v čase $t = 0$ (tedy jde o dříve zmíněnou počáteční podmínku). Získáváme tak řešení:

$$N(t) = N_0 \cdot (1 + a - b)^t \tag{3.7}$$

a tedy explicitní vyjádření velikosti populace $N(t)$, ke kterému jsme bezpochyby dospěli i při ručním výpočtu bez využití software. Obdobným způsobem vyřešíme symbolicky rovnici spojitého modelu.

Spojitý případ

Stejně jako v diskrétním případě, také v rovnici 3.5 vystupují proměnné a, b a čas t a časově závislá funkce $N(t)$. Provedeme proto obdobné kroky, nicméně pro řešení diferenciální rovnice zavoláme tentokrát příkaz `dsolve()` (předchozí kroky spojené s instalací knihovny zůstávají beze změny):

```
py_run_string("from sympy.abc import a,b,t")
py_run_string("N=Function('N')")
py_run_string("f=Derivative(N(t),t)-(a-b)*N(t)")
py_run_string("print(dsolve(f,N(t)))")
```

nyní získáváme řešení ve tvaru `C1*exp(t*(a - b))`, které se od diskrétního případu liší indexem u konstanty odpovídající počáteční podmínce (nyní jde o `C1`) a namísto operátoru pro umocnění dostáváme exponenciální funkci `exp()`. Snadno tedy provedeme interpretaci:

$$N(t) = N_0 \cdot e^{t \cdot (a-b)} \tag{3.8}$$

a tedy explicitní vyjádření velikosti populace $N(t)$ se spojitým časem (zde e je základ přirozeného logaritmu, známé *Eulerovo číslo*).

3.1.3 Analýza řešení

Nyní můžeme přistoupit k analýze výsledků řešení modelu. Využijeme k tomu grafické možnosti prostředí R.

Diskrétní případ

Abychom mohli vykreslit řešení v diskrétním případě, je nutné přiřadit parametrům modelu N_0, a a b konkrétní numerické hodnoty. Budeme proto pro demonstraci uvažovat

populaci zajíce polního, pro kterou umíme parametry zjistit. Z literatury [23] zjistíme, že samice zajíce polního vrhá 3–4 krát do roka 1–7 mláďat a že se zajíc polní dožívá 10–12 let.

Na základě těchto údajů odhadneme hodnoty koeficientů porodnosti a úmrtnosti. Předpokládáme přitom, že samic je v populaci stejný počet jako samců a plození nových jedinců je průměrné. Samice zajíce tedy 3,5 krát v roce vrhne 4 mláďata. To je celkem 14 nově narozených jedinců za rok. Jelikož je v populaci samic polovina, poměr nově narozených vzhledem k celkovému počtu jedinců populace bude „pouze“ 7. Jak zjistíme dále, tato hodnota je velmi vysoká. Proto pro větší přehlednost přejdeme k délce kroku rovné jednomu měsíci. Koeficient porodnosti tak bude dvanáctina z původně vypočítané hodnoty, tj. $a = \frac{7}{12}$.

Podobně odvodíme koeficient úmrtnosti. Zajíc polní se dožívá průměrně 11 let, za rok tedy zemře $\frac{1}{11}$ jeho populace. Opět přejdeme k měsíčnímu kroku a získáme $b = \frac{1}{11 \cdot 12}$. Nechť počáteční velikost populace N_0 je 100 jedinců. Pro časový krok rovný jednomu měsíci celkem máme:

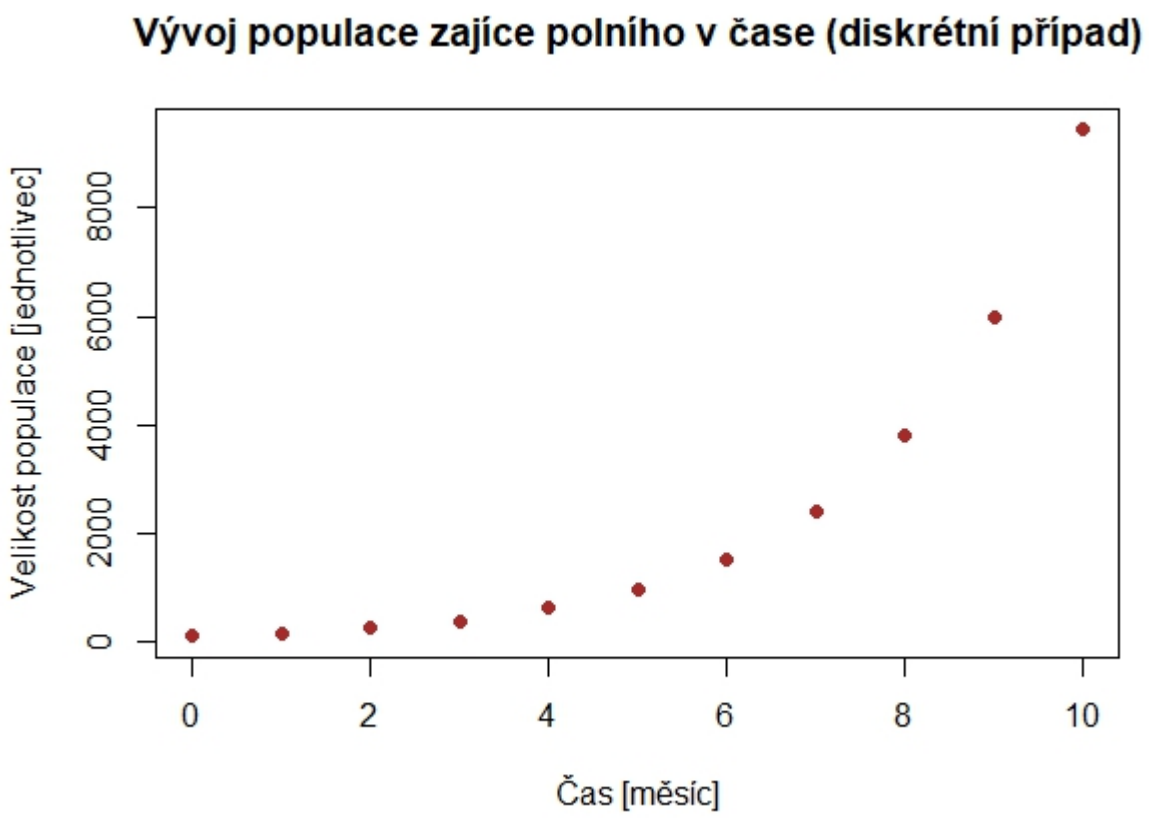
$$N_0 = 100, \quad a = \frac{7}{12}, \quad b = \frac{1}{11 \cdot 12}.$$

Řešení vykreslíme následujícím postupem. Z rovnice 3.7, která je řešením diferenční rovnice 3.1, vypočítáme hodnoty řešení pro časové body, které nás zajímají (např. $t = 1, 2, ..., 10$), a příkazem `plot()` pro vykreslování grafů je zobrazíme. Příkazu `plot()` dále nastavíme několik nepovinných argumentů (pomocí `type` určíme, že půjde o bodový graf, `pch` (point character) nastavíme na 19 pro plné puntíky, dále nastavíme `col` (color) pro barvu bodů, `main` pro nadpis grafu, `xlab` a `ylab` pro popisky os). Zdrojový kód pro vykreslení následuje a výsledek je vidět na obrázku 3.1

```
N0<-100
a <-7/12
b <-1/(11*12)

t<-0:10

reseni<-N0*(1+a-b)^t

plot(t,
     reseni,
     type="p",
     pch=19,
     col="brown",
     main="Vývoj populace zajíce polního v čase (diskrétní případ)",
     xlab="Čas [měsíc]",
     ylab="Velikost populace [jednotlivec]")
```

Spojitý případ

Využijeme nyní numerické hodnoty odpovídající populaci zajíce polního z diskrétního případu také pro případ spojitý - to nám umožní lépe porovnat obě řešení a všimnout si rozdílů mezi nimi. Vykreslení spojitých funkcí v prostředí R budeme vždy aproximovat po částech - tedy jako lomené čáry složené z úseček mezi body, které odpovídají hodnotám funkce v předem určených časových okamžicích. Výhodou je, že hustotu těchto okamžiků si můžeme volit libovolně tak, aby výsledná lomená čára poskytovala dostatečně přesnou aproximaci hladké křivky odpovídající vykreslované funkci. Nejprve si proto funkci $N(t)$ v R nadefinujeme. Jejími argumenty budou jak čas t, tak počáteční podmínky a parametry



Obrázek 3.1: Řešení diskrétního modelu pro populaci zajíce polního

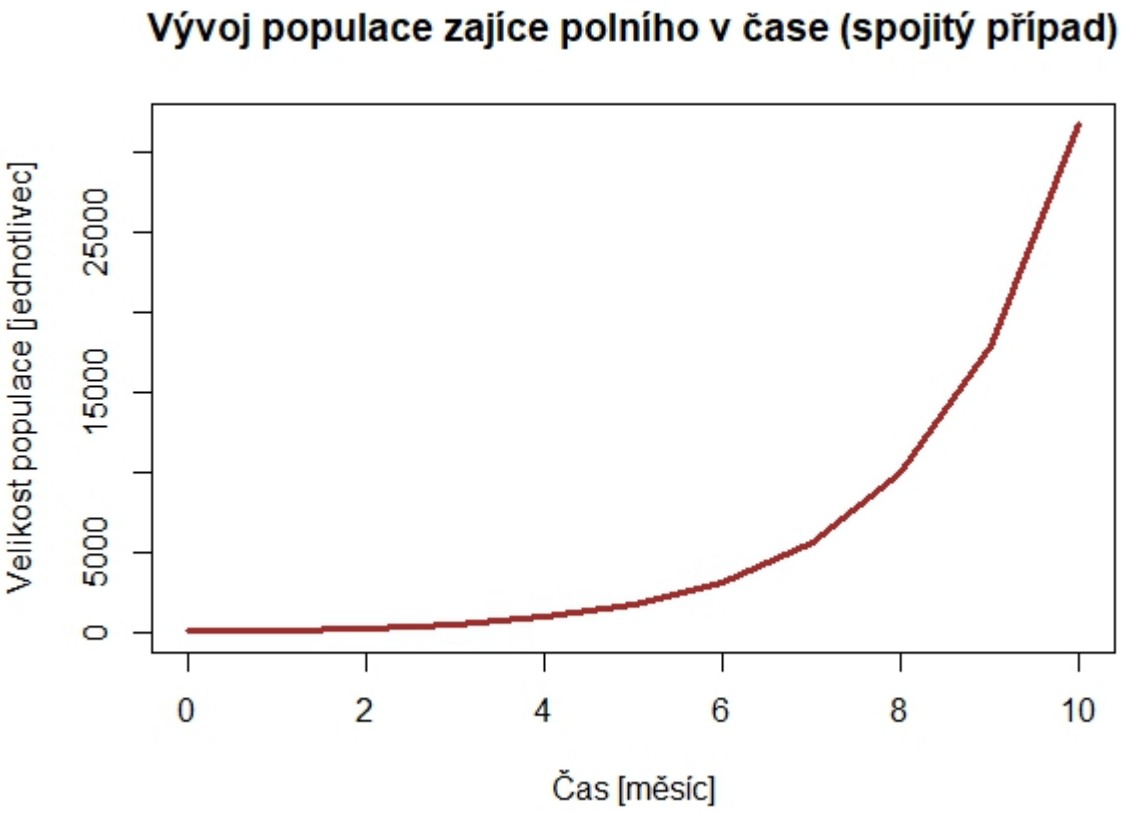
N_0, a a b. To nám umožní následně tyto hodnoty měnit, aniž bychom museli znovu definovat celou funkci $N(t)$.

```
N<-function(t,N0,a,b) {
  return(N0*exp(t*(a-b)))
}
```

Nyní můžeme vykreslit její graf s úsečkami mezi zvolenými časovými okamžiky (začněme s $t = 1, 2, ..., 10$, stejně jako v diskrétním případě). Argument `type` nyní nastavíme na `l` (liniový graf) a namísto typu bodu `pch` zvolíme šířku čáry 3 pixely pomocí argumentu `lwd`:

```
N0<-100
a <-7/12
b <-1/(11*12)

t<-0:10

plot(t,
     N(t,N0,a,b),
     type="l",
     lwd=3,
     col="brown",
     main="Vývoj populace zajíce polního v čase (spojitý případ)",
     xlab="Čas [měsíc]",
     ylab="Velikost populace [jednotlivec]")
```

Výsledný graf, viditelný na obrázku 3.2, připomíná hladkou exponenciální křivku, nicméně při pozornějším pohledu je možné rozpoznat zlomy v jedenácti předdefinovaných časových



Obrázek 3.2: Řešení spojitého modelu pro populaci zajíce polního

okamžicích. Abychom získali hladší a na pohled pěknější křivku, vrátíme se nyní v kódu o jeden krok nazpět a nadefinujeme desetkrát jemnější dělení časového intervalu:

```
t<-seq(0,10,by=0.1)
```

Po opětovném vykreslení grafu je již lomená čára (nyní se 101 zlomy) k nerozeznání od hladké exponenciální křivky, která odpovídá naší spojité funkci $N(t)$.

Pusťme se nyní do malého experimentování s nastavením koeficientů porodnosti, úmrtnosti a s počátečními podmínkami modelu. Hodnoty lze nastavit před vykreslením modelu a opakovaným spuštěním výše uvedeného kódu vykreslovat nové grafy a diskutovat rozdíly mezi výsledky. Různé variace N_0 na první pohled nemají vliv na tvar křivky modelu: v závislosti na počáteční velikosti se mění pouze výsledný počet jedinců, ovšem dynamika populace je stejná. Jedinou výjimkou je nastavení $N_0 = 0$, při kterém se (dle očekávání) nulová počáteční populace není schopná rozmnožovat a počet jedinců je stále nulový.

Jiného výsledku dosáhneme, pokud budeme pracovat se vzájemným poměrem koeficientů porodnosti a úmrtnosti a a b. Opět budeme očekávat, že pokud je porodnost vyšší než úmrtnost, populace poroste (a jak je patrné z našich dosavadních experimentů, exponenciálně), ověříme nyní také přirozená očekávání, že pokud jsou oba koeficienty stejně velké, velikost populace se nebude měnit a navíc, pokud koeficient úmrtnosti převýší koeficient porodnosti, populace po nějaké době vymře.

Zvolíme obdobný postup jako v předchozím případě, s tím rozdílem, že nejprve vykreslíme prázdný graf (v grafu bude nejprve jen jediný bod v počátku, pomocí argumentu `cex` mu nastavíme nulovou velikost, aby nebyl viditelný, dále nastavíme pomocí argumentů `xlim` a `ylim` vodorovný a svislý rozměr grafu) a do něj pak tři varianty řešení modelů s

hodnotami po řadě $a \in [0,1; 0,2; 0,3]$ a $b \in [0,3; 0,2; 0,1]$ s různými barvami čar[18].

```
# Prázdný graf (bod [0,0] a nulová velikost puntíku díky cex=0)
plot(0,
     0,
     cex=0,
     xlim=c(0,10),
     ylim=c(0,800),
     main="Vývoj populace s různými variantami koeficientů a a b",
     xlab="Čas [měsíc]",
     ylab="Velikost populace [jednotlivec]")

t<-seq(0,10,by=0.1)
N0<-100

# Cyklus se třemi variantami
for (i in 1:3) {
  a <-c(0.1,0.2,0.3)[i]
  b <-c(0.3,0.2,0.1)[i]

  lines(t,
        N(t,N0,a,b),
        lwd=3,
        col=c("forestgreen","gold","brown")[i])
}
```

Výsledek je patrný z obrázku 3.3, kde hnědá čára odpovídá rostoucí populaci ($a > b$), žlutá čára populaci z neměnnou velikostí ($a = b$) a zelená čára populaci, která postupně vymře ($a < b$). Můžeme tedy dospět k následujícím závěrům. Pokud bude koeficient porodnosti vyšší než koeficient úmrtnosti, velikost populace bude exponenciálně růst. Pokud budou hodnoty koeficientů totožné, populace bude mít stále stejnou velikost. A jestliže bude koeficient úmrtnosti vyšší než koeficient porodnosti, populace bude vymírat. Počáteční velikost populace nemá na zmíněné chování řešení žádný vliv (pokud je vyšší než nula). V diskrétním případě je situace prakticky totožná. Řešení se chová kvalitativně stejně, „malé“ rozdíly můžeme pozorovat v hodnotách velikosti populace v jednotlivých časových bodech.

3.1.4 Numerické řešení

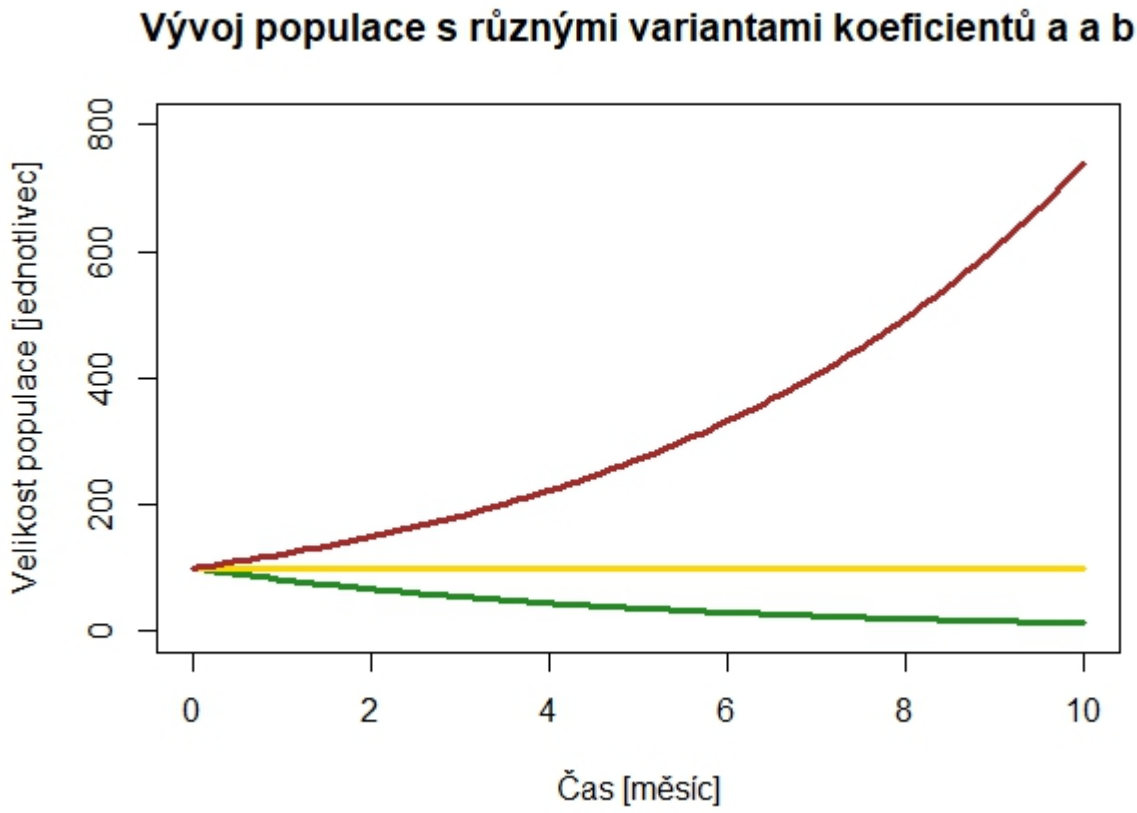
Doposud jsme pracovali se symbolickými řešeními modelů, získanými buď za pomoci ručního výpočtu nebo software Python. Uveďme si nyní, jak implementujeme vytvořený model v jazyce R a jak získáme jeho numerické řešení, které nám rovněž poslouží k vizualizaci výsledku,v některých případech dokonce jednodušší, než jsme dosud viděli.

Pro matematické modelování spočívající v řešení (soustav) diferenčních a diferenciálních rovnic v prostředí R budeme využívat tzv. řešič neboli funkci `ode()` (ordinary differential equations) z knihovny `deSolve` [32], volně dostupné na úložišti CRAN[19]. Před použitím budeme muset opět balík nainstalovat do počítače (stačí pouze jednou) a načíst (vždy při spuštění skriptu):

```
install.packages("deSolve")
library("deSolve")
```

[18] Zde zvolené barvy , a jsou víceméně náhodně vybrány z široké palety barev, které nabízí prostředí R pod jejich anglickými názvy. Alternativně lze použít zadání barvy pomocí funkce `rgb()`

[19] CRAN je zkratka pro Comprehensive R Archive Network, jde o otevřený repozitář obsahující desetitisíce knihoven pro jazyk R, které lze snadno instalovat přímo z prostředí Rstudia.



Obrázek 3.3: Řešení modelu pro různé poměry koeficientů porodnosti a a úmrtnosti b

Pro zadání (soustavy) diferenčních nebo diferenciálních rovnic, které budeme řešit, do funkce `ode()` bude vždy zapotřebí nadefinovat 5 argumentů funkce: `y`, `times`, `func`, `parm` a `method`. Argumenty nyní jen stručně představíme a podrobnější diskuzi o jejich vhodném nastavení ponecháme na později, do textu jednotlivých příkladů:

`y` ... počáteční podmínky (tedy v čase počátku řešení, obvykle v $t = 0$)
jako vektor (`c()`) rovností (blíže viz další text)

`times` ... numericky zadané hodnoty času, ve kterých chceme získat výsledek,
první hodnota je považována za počátek pro řešení

`func` ... představuje samotnou soustavu diferenciálních rovnic, soustava se zadává jako seznam (`list()`) pravých stran těchto rovnic v podobě funkcí (`function()`), jejichž argumenty jsou čas, seznam proměnných a seznam parametrů
(blíže viz další text)

`parms` ... všechny ostatní parametry vystupující v jednotlivých rovnicích
(tedy všechna čísla, která nejsou proměnné, v našem případě např. porodnost)

`method` ... metoda, kterou řešič použije
(v našem případě `lsoda` pro diferenciální a `iteration` pro diferenční rovnice)

Diskrétní případ

V případě numerického řešení v prostředí R není možné získat výsledek bez předchozího zadání konkrétních číselných hodnot všech parametrů a počátečních podmínek modelu. Můžeme proto nyní setrvat u předešlého příkladu s populací zajíce polního a podívat se, jak dospět k totožnému řešení pomocí numerických metod.

V první řadě určíme hodnoty počátečních podmínek a parametrů modelu (to provedeme zcela totožně s předchozím příkladem) a následně nadefinujeme výše uvedených pět argumentů funkce `ode()` v pořadí, ve kterém byly zmíněny výše. Argumentu `y` přiřadíme vektor počátečních podmínek `podminky`. Vektor je vhodné zadávat jako tzv. vektor s pojmenovanými prvky (named vector), tj. ve tvaru, kdy každý prvek je zadán rovnicí, na jejíž levé straně je název prvku (v našem případě velikost populace $N(t)$ nazveme prostě `N`) a na pravé straně hodnota (tedy N_0 neboli `N0`). Zadání časových okamžiků pro získání řešení zůstává stejné jako v předešlých příkladech, za pozornost ovšem stojí zadání soustavy diferenčních rovnic (v našem případě zatím jediné rovnice): soustavu zadáme jako funkci času, proměnných a parametrů, kde proměnné i parametry budou reprezentovány ve formě vektorů (opět je vhodné volit vektory s pojmenovanými prvky). Návratová struktura této funkce (argument příkazu `return()`) bude typu seznam (`list`). Každý prvek návratového seznamu obsahuje pravou stranu jedné diferenciální/diferenční rovnice na jejíž levé straně je buď derivace dané funkce nebo $n+1$. prvek dané posloupnosti. Protože naším cílem je nalézt řešení rovnice 3.1, bude mít pravá strana tvar $N(t) + N(t) \cdot (a - b)$. Vektory proměnných a parametrů je jako argumenty funkce `soustava` zapotřebí nějak pojmenovat, pro stručnost volíme názvy `prom` a `par`. Při zadávání rovnic pak jednotlivé proměnné a parametry ve vektorech indexujeme buď číselně (to je ale obvykle nepřehledné) nebo, pokud jsme zvolili pojmenování prvků, pomocí jejich názvů. Zatímco vektor parametrů specifikujeme v dalším kroku, vektor proměnných již není třeba definovat, protože informace o proměnných je již vložena do dříve nadefinovaného vektoru počátečních podmínek. Opět je vhodné vektor parametrů definovat jako vektor s pojmenovanými prvky. Posledním argumentem je metoda, kterou pro diferenciální rovnice vždy nastavíme na hodnotu `iteration`:

```
N0<-100
a <-7/12
b <-1/(11*12)

podminky<-list(N=N0)
t<-0:10
soustava<-function(t,prom,par) {
  return(list(c(prom[1]+prom[1]*(par[["a"]]-par[["b"]]))))
}
parametry<-list("a"=a,"b"=b)
metoda<-"iteration"
```

Jakmile je tímto způsobem nadefinováno všech 5 povinných argumentů, lze přikročit k nalezení samotného řešení pomocí příkazu `ode()`. Než si ovšem řešení uložíme do proměnné `reseni`, provedeme ještě jeden krok, spočívající v konverzi matice výsledků na datovou tabulku (`data.frame`), se kterou se nám bude podstatně lépe pracovat. K tomu využijeme příkazu `as.data.frame()`:

```
reseni<-as.data.frame(ode(podminky,t,soustava,parametry,metoda))
```

Vykreslení do grafu pak provedeme téměř totožně s předchozím příkladem, jen jako první dva argumenty příkazu `plot()` použijeme první dva sloupce datové tabulky `reseni` (konkrétně `reseni$time` pro čas a `reseni$N` pro velikost populace). Výsledkem pak bude obrázek totožný s obrázkem 3.1 (pro úsporu místa obrázek neopakujeme podruhé):

```
plot(reseni$time,
     reseni$N,
     type="p",
     pch=19,
     col="brown",
     main="Vývoj populace zajíce polního v čase (diskrétní případ)",
```

```
     xlab="Čas [měsíc]",
     ylab="Velikost populace [jednotlivec]")
```

Spojitý případ

Numerické řešení se ve spojitém případě výrazně neodchyluje od řešení diskrétního. Ponecháváme beze změny počáteční podmínky, nastavení i definici parametrů modelu a změny dostojí pouze argument `method`, který nastavíme na `lsoda`, což odpovídá algoritmu, který umí vybrat nejvhodnější metodu pro řešení dané soustavy diferenciálních rovnic[20]. Změny dozná též argument `soustava`, který se nyní bude skládat z pravé strany rovnice 3.5:

```
N0<-100
a <-7/12
b <-1/(11*12)

podminky<-c("N"=N0)
t<-0:10
soustava<-function(t,prom,par) {
  return(list(c(prom["N"]+prom["N"]*(par["a"]-par["b"]))))
}
parametry<-c("a"=a,"b"=b)
metoda<-"iteration"

reseni<-as.data.frame(ode(podminky,t,soustava,parametry,metoda))

plot(reseni$time,
     reseni$N,
     type="p",
     pch=19,
     col="brown",
     main="Vývoj populace zajíce polního v čase (diskrétní případ)",
     xlab="Čas [měsíc]",
     ylab="Velikost populace [jednotlivec]")
```

Stejně jako v diskrétním případě vykreslí příkaz `plot()` graf totožný s grafem, který jsme získali pro spojité řešení v předchozí kapitole. Ušetříme proto místo a odkazujeme na obrázek 3.2.

3.1.5 Modifikace modelu

Nyní porovnáme výsledky modelu s pozorováním a kriticky zhodnotíme model. Je zřejmé, že problém růstu populace je složitější, než jak jsme si jej namodelovali. Provedeme proto vylepšení modelu, které jej více přiblíží realitě.

Velikost populace se zřejmě nemůže exponenciálně zvyšovat do nekonečna. Prostor, v němž populace žije, je omezený, podobně jako množství živin, které má k dispozici. Doplňme proto předpoklad modelu, že úmrtnost se bude zvyšovat se zvětšující se populací. Nejjednodušší způsob závislosti je lineární závislost. Koeficient úmrtnosti tedy nebudeme již chápat jako konstantní číslo, ale jako rostoucí lineární funkci času $b + c \cdot N(t)$, kde b a c jsou reálná nezáporná čísla.

[20] Nastavení argumentu `method="lsoda"` volá poměrně letitou knihovnu lsoda napsanou v jazyce Fortran, která obsahuje algoritmy, jež se snaží výběrem vhodné metody předejít tzv. tuhému (anglicky stiff) problému, tj. nestabilitě při hledání řešení soustavy diferenciálních rovnic. Nejde tedy přímo o metodu řešení, ale jakousi „nadstavbu“, která provede výběr za nás.

Podobně jako dříve získáme dvě rovnice modelu.

Diskrétní případ:

$$N(t+1) = (1+a-b) \cdot N(t) - c \cdot N(t)^2, \quad N(0) = N_0 \tag{3.9}$$

Spojitý případ:

$$N'(t) = (a-b) \cdot N(t) - c \cdot N(t)^2, \quad N(0) = N_0 \tag{3.10}$$

V diskrétním případě se nám nepodaří získat analytické řešení modelu jako dříve, z rekurentního předpisu však můžeme určit velikost populace v libovolném čase t. Zaměřme se proto nejprve na spojitý případ, v němž je analýza řešení jednodušší.

Jak jsme přislíbili v předchozím textu, ještě jednou (naposledy) zavoláme na pomoc knihovnu `sympy` jazyka Python, zprostředkovanou knihovnou `reticulate` jazyka R. Po správném zadání tak obdržíme řešení diferenciální rovnice 3.10:

```
py_run_string("from sympy.abc import a,b,c,t")
py_run_string("N=Function('N')")
py_run_string("f=Derivative(N(t),t)-(a-b)*N(t)+c*N(t)**2")
py_run_string("print(dsolve(f,N(t)))")
```

Výsledek `(a-b)*exp(a*(C1+t))/(c*(exp(a*(C1+t))-exp(b*(C1 + t))))` můžeme tentokrát interpretovat jako

$$N(t) = \frac{(a-b) \cdot e^{a \cdot (C_1+t)}}{c \cdot \left[e^{a \cdot (C_1+t)} - e^{b \cdot (C_1+t)}\right]} \tag{3.11}$$

nicméně konstanta C_1, kterou jsme tímto způsobem získali, se nám interpretuje hůře, než dříve používaná počáteční velikost populace N_0. Z dosazení $t = 0$ nicméně můžeme úpravou získat vztah pro substituci:

$$C_1 = \frac{ln(1 - \frac{(a-b)}{N_0 \cdot c})}{b-a}$$

a po dosazení konečně předpis funkce $N(t)$ obsahující počáteční velikost populace N_0:

$$N(t) = \frac{N_0 \cdot (a-b)}{N_0 \cdot c + (a-b-N_0 \cdot c) \cdot e^{(b-a)t}} \tag{3.12}$$

Přesto, že formulace modelu 3.10 byla relativně jednoduchá, vidíme, že získané řešení je tvořeno funkcí s poměrně složitým předpisem. Skutečně, úroveň složitosti symbolických řešení roste se zvyšující se komplexitou předpisu modelu poměrně rychle a již poměrně jednoduše formulované modely mohou vést na soustayv rovnic, které vůbec nejsme schopni řešit symbolicky[21]. Proto se v následujícím textu omezíme již pouze na numerická řešení získaná pomocí řešiče `ode()` v prostředí R a symbolickými řešeními se nadále nebudeme zabývat.

Analýzu řešení provedeme znovu graficky, za použití příkazu `plot`. Přepíšeme získané řešení do prostředí R a opakovaně budeme měnit parametry a, b a c a počáteční podmínku

[21] Důvodem je obvykle neznalost algoritmu pro řešení daného typu soustav diferenciálních rovnic. Je proto možné, že symbolické řešení existuje, ale nikdo dosud neobjevil nástroje pro jeho nalezení.

$N_0 = 100$ a zkoumat závislost řešení na jejich hodnotách. Na základě dříve uvedených údajů zvolme: $N_0 \in [50; 500]$, $a \in [0; 1,17]$, $b, c \in [0,007; 0,0083]$ (připustíme, že koeficient porodnosti může být i nulový, abychom mohli pozorovat vývoj řešení i v případě, kdy je koeficient porodnosti nižší než koeficient úmrtnosti). Velikost koeficientu c pro jednoduchost považujme za srovnatelnou s koeficientem b. Graf řešení odpovídajícího následujícímu kódu pro původní hodnoty je vykreslen na obrázku 3.4.

```
N<-function(t,N0,a,b) {
  return((N0*(a-b))/(N0*c+(a-b-N0*c)*exp((b-a)*t)))
}

N0<-100
a <-7/12
b <-1/(11*12)
c <-1/(11*12)

t<-seq(0,10,by=0.1)

plot(t,
     N(t,N0,a,b),
     type="l",
     lwd=3,
     col="brown",
     main="Modifikovaný vývoj populace zajíce (spojitý případ)",
     xlab="Čas [měsíc]",
     ylab="Velikost populace [jednotlivec]")
```

Nastavováním různých hodnot parametrů v rámci uvedených rozsahů zjistíme, že velikost populace se vždy (po „dostatečně“ dlouhém čase) ustálí na nějaké hodnotě, při níž je populace v dynamické rovnováze s prostředím (neroste ani nevymírá). V R můžeme tuto hodnotu určit snadno dosazením hodnoty $t = \infty$ do funkce $N(t)$, (tedy spuštěním kódu `N(Inf,N0,a,b)`), obecněji však bude vhodné ji určit z rovnice modelu položením levé strany rovné nule, tj. vynucením nulové derivace (změny) velikosti populace v čase:

$$0 = (a - b) \cdot N(t) - c \cdot N(t)^2 \tag{3.13}$$

Rovnice 3.13 má jediné nenulové řešení, a to:

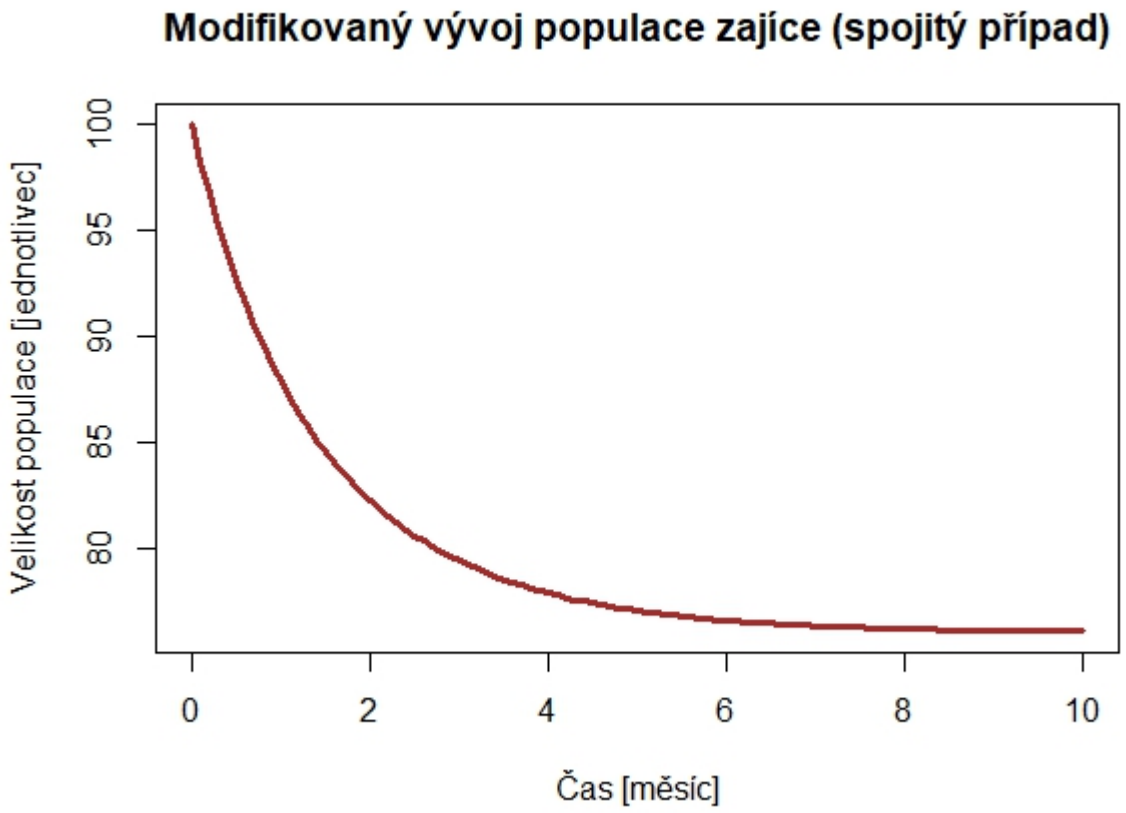
$$N(t) = \frac{a - b}{c}.$$

Pokud $a \geq b$, velikost populace se ustálí na hodnotě $\frac{a-b}{c}$ (tj. $\lim_{t\to\infty} N(t) = \frac{a-b}{c}$). Pokud $a < b$, populace vymře (stejně jako v předcházejícím modelu). Zmíněné chování platí pro libovolnou nenulovou počáteční velikost populace.

Právě odvozenou ustálenou hodnotu velikosti populace $\frac{a-b}{c}$ nazýváme *úživností* (resp. *kapacitou*) *prostředí* a značíme písmenem K. Označme dále

$$r = a - b.$$

Tento parametr, tj. rozdíl koeficientů porodnosti a úmrtnosti, se nazývá *vnitřní koeficient růstu populace*.



Obrázek 3.4: Řešení modifikovaného modelu pro hodnoty $a = \frac{7}{12}$, $b = \frac{1}{11 \cdot 12}$, $c = \frac{1}{11 \cdot 12}$ *a* $N_0 = 100$

Při tomto označení můžeme spojitý případ rovnice původního modelu 3.5 přepsat na tvar:

$$N'(t) = r \cdot N(t) \tag{3.14}$$

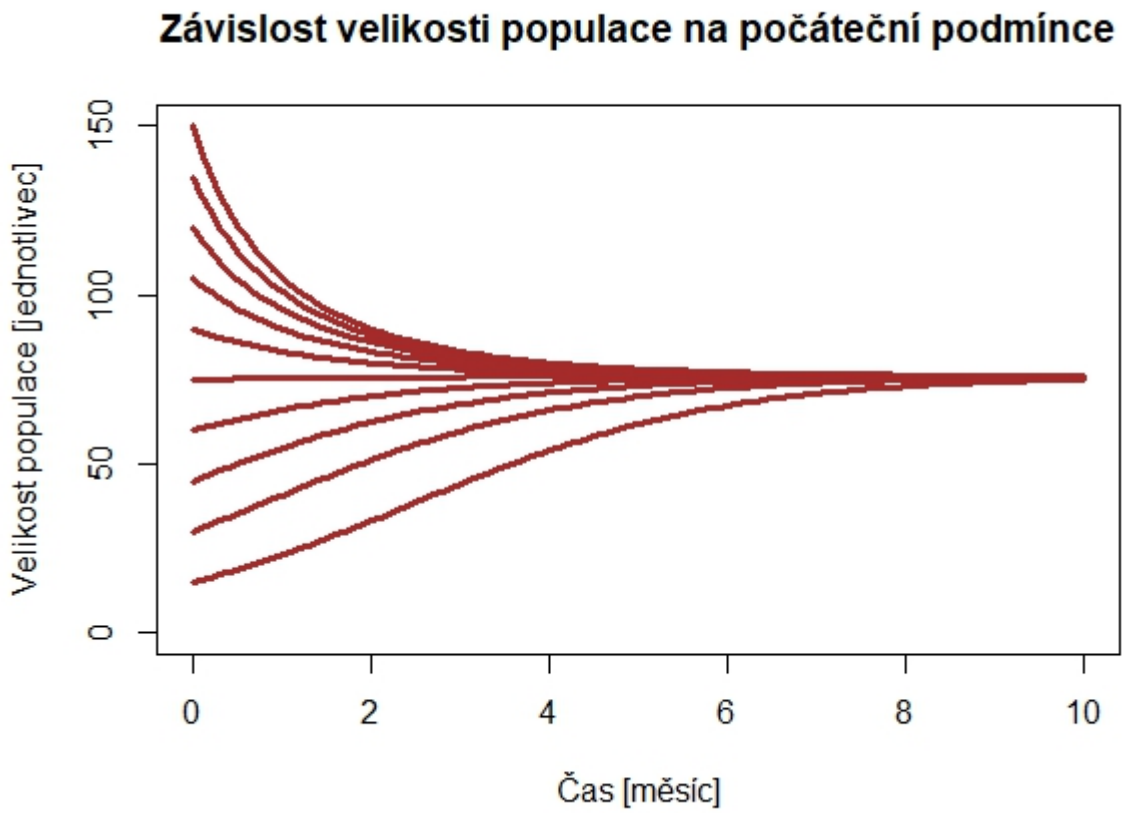
a spojitý případ rovnice modifikovaného modelu na tvar:

$$N'(t) = r \cdot \left(1 - \frac{N(t)}{K}\right) \cdot N(t). \tag{3.15}$$

Chování řešení v případě modifikovaného modelu závisí na počáteční velikosti populace. Velikost populace v čase klesá, pokud $N(0) > K$, roste, pokud $N(0) < K$, případně zůstává konstantní. Závislost řešení na počáteční velikosti populace dokumentuje následující kód a obrázek 3.5, v němž $K = 76$ a $N_0 \in \{15; 30; \ldots; 150\}$. Pomocí for-cyklu je vygenerováno 10 řešení pro jednotlivá N_0, která jsou vykreslena v jednom grafu příkazem `lines()`.

```
N<-function(t,N0,K,r) {
  return(1/(1/K+(1/N0-1/K)*exp(-r*t)))
}

r<-76/(11*12)
K<-76
```

```
t<-seq(0,10,by=0.1)

plot(0,
     0,
     cex=0,
     xlim=c(0,10),
     ylim=c(0,150),
     main="Závislost velikosti populace na počáteční podmínce",
     xlab="Čas [měsíc]",
     ylab="Velikost populace [jednotlivec]")

for(i in 1:10) {
  N0<-15*i
  lines(t,N(t,N0,K,r),lwd=3,col="brown")
}
```



Obrázek 3.5: Řešení modifikovaného modelu pro hodnoty $a = \frac{7}{12}$, $b = \frac{1}{11 \cdot 12}$, $c = \frac{1}{11 \cdot 12}$ *a* $N_0 \in \{15; 30; \ldots; 150\}$

Vraťme se nyní k diskrétnímu případu modifikovaného modelu 3.9, který je možné po zavedení parametrů K a r přepsat do tvaru:

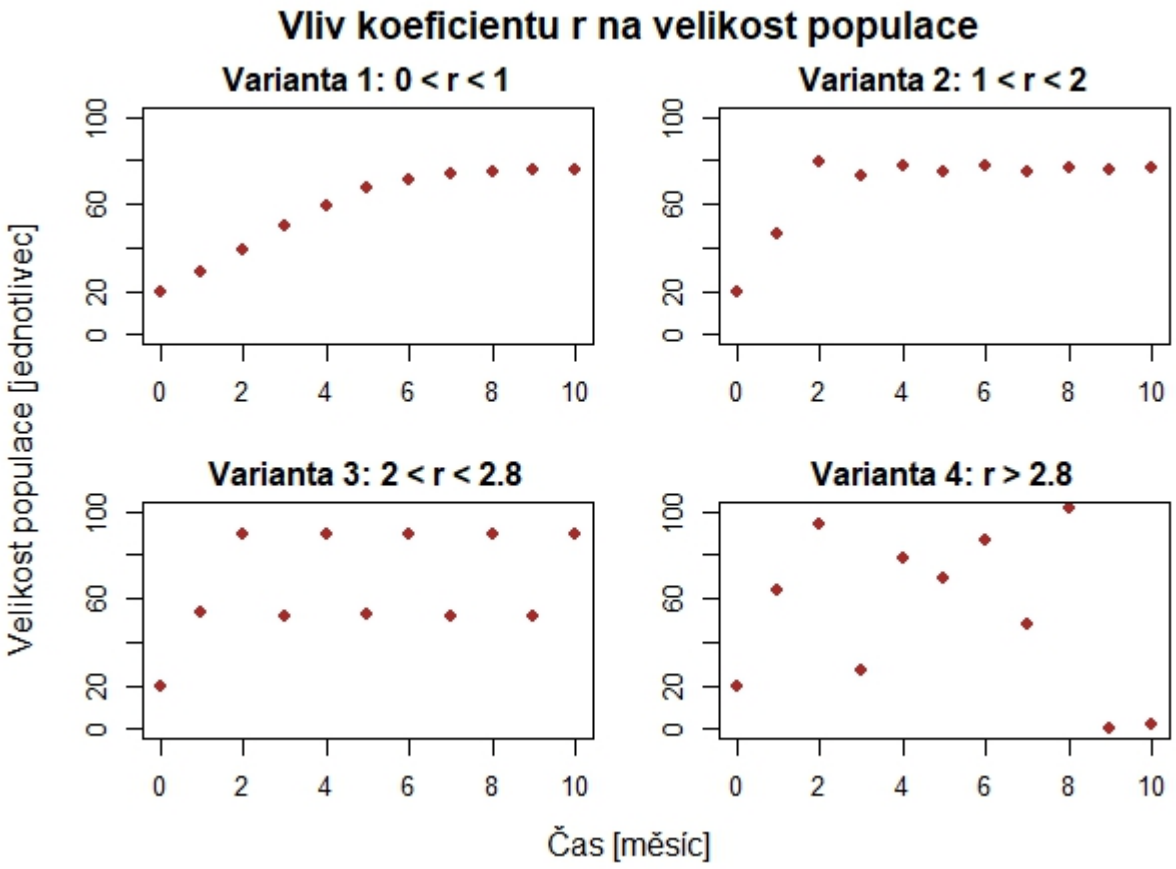
$$N(t+1) = N(t) + r \cdot N(t) \cdot \left(1 - \frac{N(t)}{K}\right), \quad N(0) = N_0. \tag{3.16}$$

Analýza diskrétních případů populačních modelů bývá složitější. Nejinak je tomu v tomto případě. Pro jednoduchost uvažujme situaci, kdy je nenulová počáteční velikost populace

N_0 nižší než úživnost prostředí K pro tuto populaci. V závislosti na velikosti vnitřního koeficientu růstu r můžeme pozorovat čtyři (kvalitativně) různá řešení (viz [28]): velikost populace monotónně roste k hodnotě úživnosti prostředí (varianta 1), přibližuje se k ní s tlumenými oscilacemi (varianta 2), kolísá kolem této hodnoty pravidelně (varianta 3), nebo kolem této hodnoty kolísá nepravidelně, chaoticky (varianta 4). Dříve použitým hodnotám parametrů r a K (u spojitého případu) odpovídá první varianta (obrázku 3.6).

Před vykreslením obrázků spočteme hodnoty posloupnosti $N(t)$ pro všechny čtyři varianty vnitřního koeficientu růstu r (označme jej r_1, r_2, r_3 a r_4), pomocí jednoduchého `for` cyklu, tedy rekurentním způsobem. Vyhneme se tak nutnosti vyřešit model symbolicky a získáme přesné řešení. Využijeme pro to přirozenou vlastnost jazyka R, který nám umožňuje jednoduše pracovat s vektory a datovými tabulkami a nadefinujeme r jako vektor o čtyřech prvcích a N jako datovou tabulku o čtyřech sloupcích. Pro vykreslení čtveřice grafů využijeme dále následující argumenty grafického příkazu `par()`: `mfrow` pro nastavení počtu grafů nad sebou a vedle sebe, `mar` pro nastavení okrajů grafu a `oma` pro nastavení okrajů celého obrázku:

```
r<-c(76/(11*12),1.8,2.3,3.0)
N0<-20
K <-76

t<-0:10
popisky<-c("0 < r < 1","1 < r < 2","2 < r < 2.8","r > 2.8")

N<-as.data.frame(matrix(NA,11,4))

par(mfrow=c(2,2),oma=c(1.5,1.5,2,0)) # nastavení řádků a sloupců
for(var in 1:4) {
  N[1,var]<-N0
  for(cas in 1:10) {
    N[cas+1,var]<-N[cas,var]+
      r[var]*N[cas,var]*
      (1-N[cas,var]/K)
  }
  par(mar=c(3,3,2,1))
  plot(t,
       N[,var],
       type="p",
       pch=19,
       ylim=c(0,100),
       col="brown",
       main=paste0("Varianta ",var,": ",popisky[var]),
       xlab="",
       ylab="")
}
par(mfrow=c(1,1))
title(main="Vliv koeficientu r na velikost populace",
      line=0.5,outer=TRUE)
mtext("Čas [měsíc]",side=1,line=0,outer=TRUE)
mtext("Velikost populace [jednotlivec]",side=2,
      line=0.5,outer=TRUE)
```



Obrázek 3.6: Vliv vnitřního koeficientu růstu na velikost populace

3.2 Populace pod tlakem nespecializovaného predátora

3.2.1 Identifikace a sestavení modelu

V předešlém příkladu jsme modelovali růst populace, která není ovlivněna svým okolím. Nyní přejdeme k situaci, kdy je v prostředí, v němž se uvažovaná populace vyvíjí, také nespecializovaný predátor. Nespecializovaný predátor je takový, který není závislý na kořisti z uvažované populace, má i alternativní zdroje obživy. Velikost jeho populace tedy můžeme považovat za konstantní a do modelu ji nemusíme zahrnovat.

První předpoklad při tvorbě modelu bude přirozený: predátoři uloví takové množství kořisti, které je úměrné době lovu, tj. množství ulovené kořisti za časový interval délky h je rovno $p \cdot h$. Parametr p se nazývá *intenzita predace* a vyjadřuje predační tlak vyvíjený na uvažovanou populaci, přesněji řečeno: množství kořisti, které predátoři uloví za jednotku času. Intenzita predace závisí na velikosti N populace kořisti, tj. $p = p(N)$. Abychom tuto závislost vyjádřili, přijmeme další dva přirozené předpoklady:

- pokud není uvažovaná populace přítomná, predátoři nic neuloví a živí se alternativní potravou,
- pokud je uvažovaná populace veliká (větší než predátoři dokáží sníst), loví predátoři jen omezené množství jedinců, které představuje jakousi *hladinu nasycení*.

Tyto předpoklady zapíšeme ve tvaru rovností:

$$p(0) = 0, \quad p(N) = S \text{ pro } N > N_{krit},$$

kde parametr S představuje hladinu nasycení, N_{krit} kritickou hodnotu velikosti populace (je-li velikost uvažované populace větší než kritická, predátoři jsou nasycení). Zvolme za $p(N)$ nejjednodušší funkci, která splňuje uvedené předpoklady, tedy funkci lineární na intervalu $[0; N_{krit}]$, jinak konstantní:

$$p(N) = \begin{cases} S \cdot \frac{N}{N_{krit}} & \dots \quad N \leq N_{krit}, \\ S & \dots \quad N > N_{krit} \end{cases} \tag{3.17}$$

U předešlého modelu jsme v diskrétním případě dospěli k rovnici:

$$N(t+1) = N(t) + r \cdot N(t) \cdot \left(1 - \frac{N(t)}{K}\right), \quad N(0) = N_0 \tag{3.18}$$

a ve spojitém případě k rovnici

$$N'(t) = r \cdot \left(1 - \frac{N(t)}{K}\right) \cdot N(t), \quad N(0) = N_0. \tag{3.19}$$

Obě tyto rovnice lze za použití časového kroku h vyjádřit ve tvaru

$$N(t+h) = N(t) + r \cdot N(t) \cdot \left(1 - \frac{N(t)}{K}\right) \cdot h, \quad N(0) = N_0. \tag{3.20}$$

Výraz

$$N(t) \cdot \left(1 - \frac{N(t)}{K}\right) \cdot h, \quad N(0) = N_0$$

lze interpretovat jako přirozený přírůstek velikosti populace za časový interval délky h. Rovnice 3.18 je rovnicí 3.20 pro $h = 1$, rovnice 3.19 je mezním případem rovnice 3.20 pro $h \to 0$.

Nyní vyjádříme změnu velikosti populace za časový interval délky h jako přirozený přírůstek populace zmenšený o množství ulovených jedinců. Rovnici 3.20 tedy dále modifikujeme a dostaneme model růstu populace pod tlakem nespecializovaného predátora ve tvaru

$$N(t+h) = N(t) + r \cdot N(t) \cdot \left(1 - \frac{N(t)}{K}\right) \cdot h - p(N(t)) \cdot h, \quad N(0) = N_0. \tag{3.21}$$

kde funkce $p(N)$ je dána rovností 3.17. Pro $h = 1$ dostaneme model diskrétní:

$$N(t+1) = N(t) + r \cdot N(t) \cdot \left(1 - \frac{N(t)}{K}\right) - p(N(t)), \quad N(0) = N_0, \tag{3.22}$$

limitním přechodem $h \to 0$ dostaneme model spojitý:

$$N'(t) = r \cdot N(t) \cdot \left(1 - \frac{N(t)}{K}\right) - p(N(t)), \quad N(0) = N_0. \tag{3.23}$$

3.2.2 Implementace modelu a jeho řešení

S rostoucí složitostí modelu vzrůstá i náročnost výpočtu řešení. Podobně jako v předchozím modifikovaném modelu nejsme schopni určit řešení diskrétního případu 3.22, z rekurentního předpisu však můžeme (při znalosti numerických hodnot parametrů modelu) určit hodnotu velikosti populace v libovolném čase t. Ve spojitém případě je možné rovnici 3.23 rozložit na dva případy (podle 3.17) a v R již známým postupem při položení derivace $N'(t) = 0$ nalézt řešení pro každý případ zvlášť. Tento postup je však možný jedině tehdy, kdy je velikost populace stále „pod hladinou“ N_{krit}, nebo stále nad ní (připouštíme i rovnost).

V obecném případě je nutné řešit rovnici 3.23 za pomoci numerických metod, k čemuž musíme znát numerické hodnoty všech parametrů modelu. Můžeme opět uvažovat populaci zajíce polního, na jehož území nyní žije i liška obecná jakožto nespecializovaný predátor. Musíme však zjistit hodnoty použitých parametrů. Od této chvíle se pro zbytek textu odkloníme od konkrétních hodnot naměřených v praxi a pro jednoduchost a názornost budeme volit „vymyšlené“ (avšak realistické) nastavení parametrů. V grafech proto dále nebudeme uvádět jednotky (které se mohou lišit pro různé modelované populace), neboť nám jde o kvalitativní chování populací. V tuto chvíli zvolme například následující nastavení:

$$S = 300,\ \ r = 1,\ \ K = 1000,\ \ N_{krit} = 200,\ \ N_0 = 500.$$

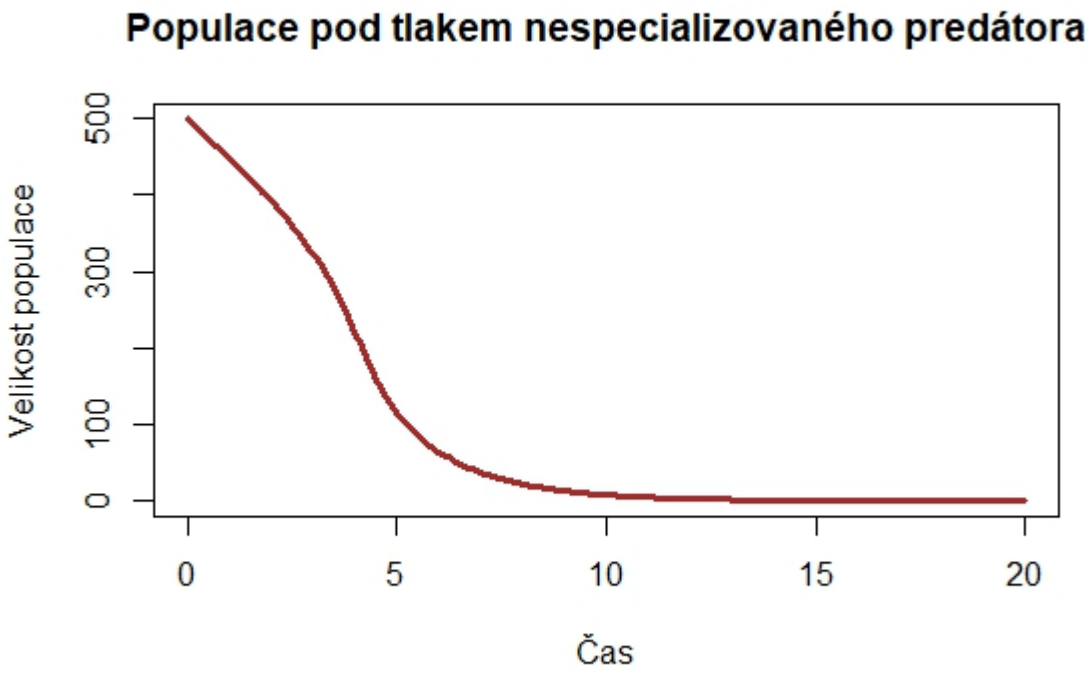
Před vlastním řešením modelu je třeba nadefinovat v R predační funkci $p(N)$ tak, jak jsme si ji popsali v 3.17. Funkce bude mít tři argumenty: velikost populace N, kritickou hladinu, při které dojde k nasycení predátora N_{krit} a vlastní hodnotu nasycení S:

```
p<-function(N,Nkrit,S) {
  if (N <= Nkrit) {
    return((S/Nkrit)*N)
  }
  else{
    return(S)
  }
}
```

Nyní lze přistoupit k nadefinování hodnot parametrů a vlastnímu výpočtu a vykreslení řešení modelu (obr 3.7:

```
N0<-500
r<-1
K<-1000
Nkrit<-200
S<-300

podminky<-c("N"=N0)
t<-seq(0,20,by=0.1)
soustava<-function(t,prom,par) {
  return(list(c(par["r"]*prom["N"]*(1-prom["N"]/par["K"])-
                  p(prom["N"],par["Nkrit"],par["S"]))))
}
parametry<-c("r"=r,"K"=K,"Nkrit"=Nkrit,"S"=S)
metoda<-"lsoda"

reseni<-as.data.frame(ode(podminky,t,soustava,parametry,metoda))

plot(reseni$time,
```

```
reseni$N,
type="l",
lwd=3,
col="brown",
main="Populace pod tlakem nespecializovaného predátora",
xlab="Čas",
ylab="Velikost populace")
```



Obrázek 3.7: Numerické řešení rovnice 3.23

Obě rovnice 3.22 a 3.23 vyjadřují změnu velikosti populace v čase. Ustálená hodnota řešení je přitom taková, která se v čase nemění, tj. v diskrétním případě pro ni platí:

$$N(t+1) - N(t) = 0 \tag{3.24}$$

a ve spojitém:

$$N'(t) = 0. \tag{3.25}$$

Obojí vede na rovnici

$$r \cdot N(t) \cdot \left(1 - \frac{N(t)}{K}\right) - p(N(t)) = 0. \tag{3.26}$$

Rovnici 3.26 nyní vyřešíme pro oba případy (viz 3.17):

1. Budeme předpokládat, že populace dravce není nasycena a tedy predační funkce $p(N)$ se nachází ve své lineárně rostoucí fázi: $p(N(t)) = S \cdot \frac{N}{N_{krit}}$. Získáváme tak kvadratickou rovnici:

$$r \cdot N(t) \cdot \left(1 - \frac{N(t)}{K}\right) - S \cdot \frac{N(t)}{N_{krit}} = 0, \tag{3.27}$$

   kterou můžeme upravit na tvar

$$N(t)^2 - \left(K - \frac{K \cdot S}{N_{krit} \cdot r}\right) \cdot N(t) = 0.$$

   Odtud ihned dostáváme dvojici řešení:

$$N_1(t) = 0, \quad N_2(t) = K \cdot \left(1 - \frac{S}{N_{krit} \cdot r}\right).$$

   První, nulové, řešení je triviální, druhé si uložíme pro další analýzu.

2. Ve druhém případě budeme předpokládat, že populace dravce je nasycena a tedy pro predační funkci platí $P(N(t)) = S$. Rovnice se nyní změní na:

$$r \cdot N(t) \cdot \left(1 - \frac{N(t)}{K}\right) - S = 0, \tag{3.28}$$

   a po úpravě

$$N(t)^2 - K \cdot N(t) + \frac{K \cdot S}{r} = 0,$$

   odkud získáváme opět dvě řešení:

$$N_3(t) = \frac{1}{2} \cdot K \cdot \left(1 - \sqrt{1 - \frac{4 \cdot S}{r \cdot K}}\right), \quad N_4(t) = \frac{1}{2} \cdot K \cdot \left(1 + \sqrt{1 - \frac{4 \cdot S}{r \cdot K}}\right).$$

Stejně jako u předchozího modifikovaného modelu je analýza diskrétní varianty modelu složitější. Opět se proto nejprve pustíme do analýzy spojitého případu, o diskrétní variantě se zmíníme později.

V závislosti na hodnotě počáteční velikosti populace a vztahu mezi parametry modelu se velikost populace v čase ustálí na jedné ze čtyř právě nalezených hodnot N_1, N_2, N_3 nebo N_4. V prvním případě populace vymře (její velikost se ustálí na hodnotě 0). Celkem je možné rozlišit 5 různých kvalitativních situací. Pro každou z nich následuje obrázek řešení s vytvořujícím kódem v prostředí R.

Pro nalezení řešení modelu je použit příkaz `ode()`, jenž zajistí numerické řešení příslušné diferenciální rovnice. Řešení je provedeno pomocí `for` cyklu vždy 21krát pro různé počáteční

hodnoty ($N_0 \in \{0, 50, \ldots, 1000\}$). Predační funkci $p(N)$ již znovu nedefinujeme, používáme výše uvedenou definici. Grafy řešení jsou vytvořeny příkazy `plot()` a `lines()` opět pomocí cyklu, obdobně jako v obrázku 3.5.

Protože kód pro výpočet je kromě části s definicí parametrů modelu S a N_{krit} stále stejný, uvedeme jej pouze jednou na začátku a nebudeme jej kromě uvedených dvou řádků neustále opakovat:

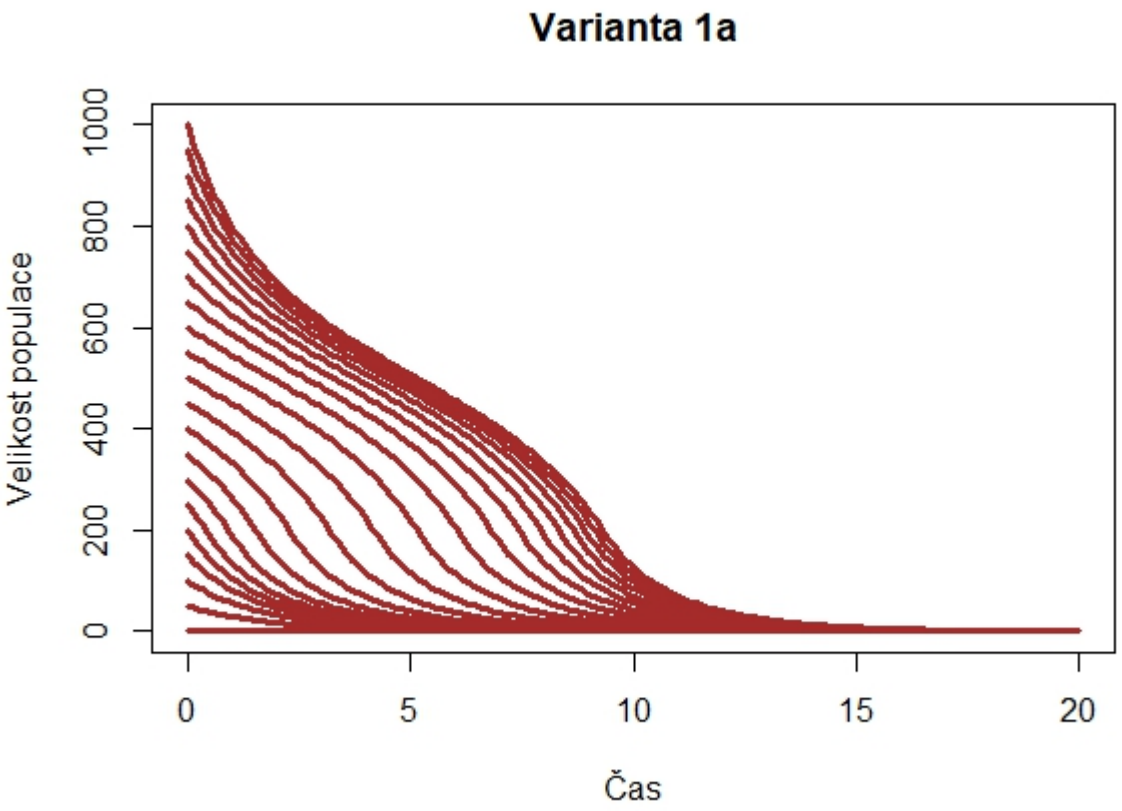
```
r<-1
K<-1000
Nkrit<-200
S<-300

t<-seq(0,20,by=0.1)
soustava<-function(t,prom,par) {
  return(list(c(par["r"]*prom["N"]*(1-prom["N"]/par["K"])-
                  p(prom["N"],par["Nkrit"],par["S"]))))
}
parametry<-c("r"=r,"K"=K,"Nkrit"=Nkrit,"S"=S)
metoda<-"lsoda"

reseni<-as.data.frame(matrix(NA,201,21))

for(i in 0:20) {
  podminky<-c("N"=50*i)

  reseni[,i+1]<-as.data.frame(ode(podminky,t,soustava,parametry,
                                  metoda))$N
}

plot(0,
     0,
     cex=0,
     xlim=c(0,20),
     ylim=c(0,1000),
     main="Varianta 1a",
     xlab="Čas",
     ylab="Velikost populace")

for(i in 0:20) {
     lines(t,reseni[,i+1],lwd=3,col="brown")
}
```

1. $S \geq \frac{1}{4} \cdot r \cdot K$

   a) $N_{krit} < \frac{S}{r}$

   Predátor vyhubí uvažovanou populaci ($N(t) = N_1$) při jakékoliv počáteční hodnotě $N_0 \geq 0$. Řešení je patrné z obrázku 3.8, který je výsledkem kódu uvedeného výše pro nastavení parametrů:

$$S = 300,\ \ r = 1,\ \ K = 1000,\ \ N_{krit} = 200,\ \ N_0 \in \{0; 50; \ldots; 1000\}$$
$$\Rightarrow N_1 = 0$$



Obrázek 3.8: Varianta 1a

b) $N_{krit} \leq \frac{S}{r}$
Velikost populace se ustálí na hodnotě N_2 při jakékoliv počáteční hodnotě $N_0 > 0$, tj. predátor zmenší ustálenou velikost populace (z hodnoty K). Nulová populace zůstává přirozeně nadále nulovou – obrázek 3.9 pro nastavení parametrů:

$$S = 300,\ \ r = 1,\ \ K = 1000,\ \ N_{krit} = 400,\ \ N_0 \in \{0; 50; \ldots; 1000\}$$
$$\Rightarrow N_1 = 0,\ \ N_2 = 250$$

Tedy v předešlém kódu v prostředí R změníme definici parametrů N_{krit} a S takto:

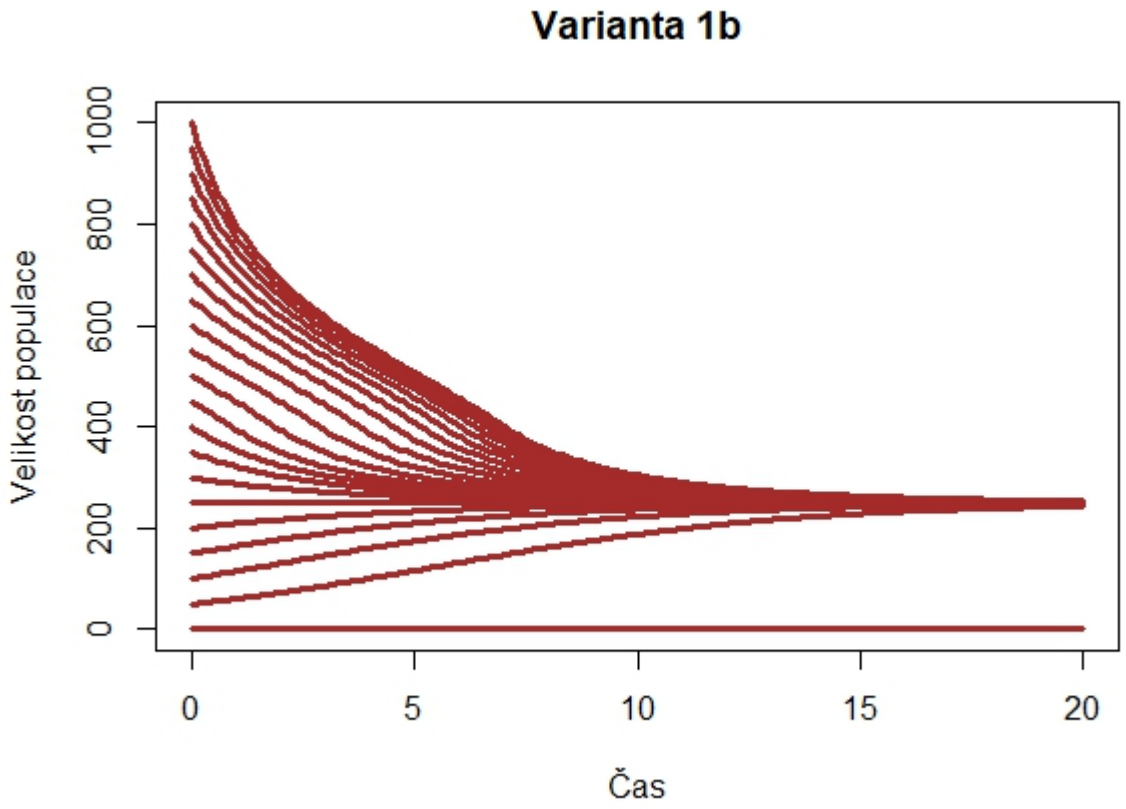
```
Nkrit<-400
S<-300
```

a získáme tak řešení $N_1 = 0$ a $N_2 = 250$.

2. $S < \frac{1}{4} \cdot r \cdot K$

a) $N_{krit} > N_3$
Velikost populace se ustálí na hodnotě N_4 při jakékoliv počáteční hodnotě $N_0 > 0$, tj. predátor zmenší ustálenou velikost populace (z hodnoty K). Nulová populace zůstává přirozeně nadále nulovou – obrázek 3.10 pro nastavení parametrů:

$$S = 200,\ \ r = 1,\ \ K = 1000,\ \ N_{krit} = 400,\ \ N_0 \in \{0; 50; \ldots; 1000\}$$
$$\Rightarrow N_1 = 0,\ \ N_4 \doteq 724$$



Obrázek 3.9: Varianta 1b

Tedy v předešlém kódu v prostředí R změníme definici parametrů N_{krit} a S takto:

```
Nkrit<-400
S<-200
```

a získáme tak řešení $N_1 = 0$ a $N_4 \doteq 724$.

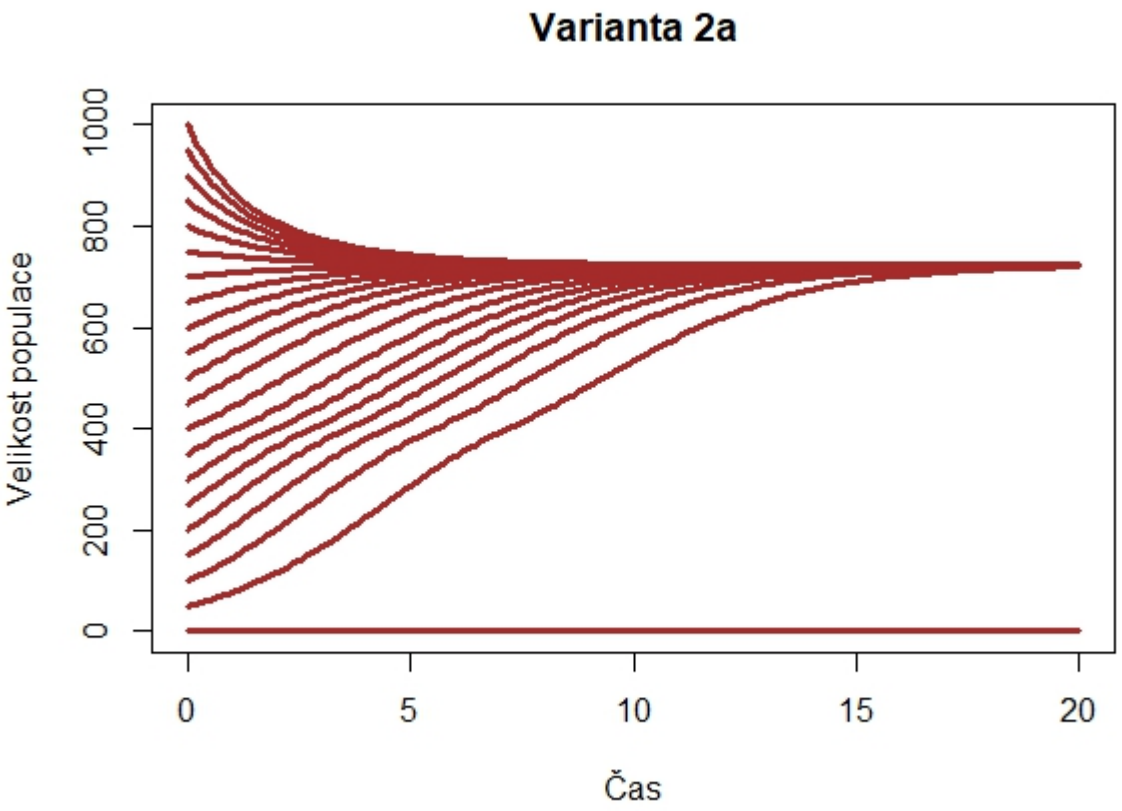
b) $\frac{S}{r} < N_{krit} \leq N_3$

Pokud $N_0 > N_3$, velikost populace se ustálí na hodnotě N_4. Pokud $N_0 < N_3$, ustálí se na hodnotě N_2. Pro $N_0 = N_3$ zůstane velikost populace stejná. Predátor tedy opět zmenší ustálenou velikost populace; nová ustálená velikost populace však závisí na její počáteční velikosti. To je patrné z obrázku 3.11 s přidanou čárkovanou čarou pro $N_0 = N_3$ při nastavení parametrů:

$$S = 200,\ \ r = 1,\ \ K = 1000,\ \ N_{krit} = 400,\ \ N_0 \in \{0; 50; \ldots; 1000\}$$
$$\Rightarrow N_1 = 0,\ \ N_2 = 200,\ \ N_3 \doteq 276,\ \ N_4 \doteq 724$$

Přidání čáry do přesné hodnoty pro N_3 provedeme následujícím kódem (s nastavením argumentu příkazu `lines()` pro typ čáry `lty=5` neboli dlouhé čárkování):

```
# Přidání řešení N3:
podminky<-c("N"=K/2*(1-sqrt(1-4*S/(r*K))))
reseni[,22]<-as.data.frame(ode(podminky,t,soustava,parametry,
                               metoda))$N
lines(t,reseni[,22],lwd=3,lty=5,col="brown")
```



Obrázek 3.10: Varianta 2a

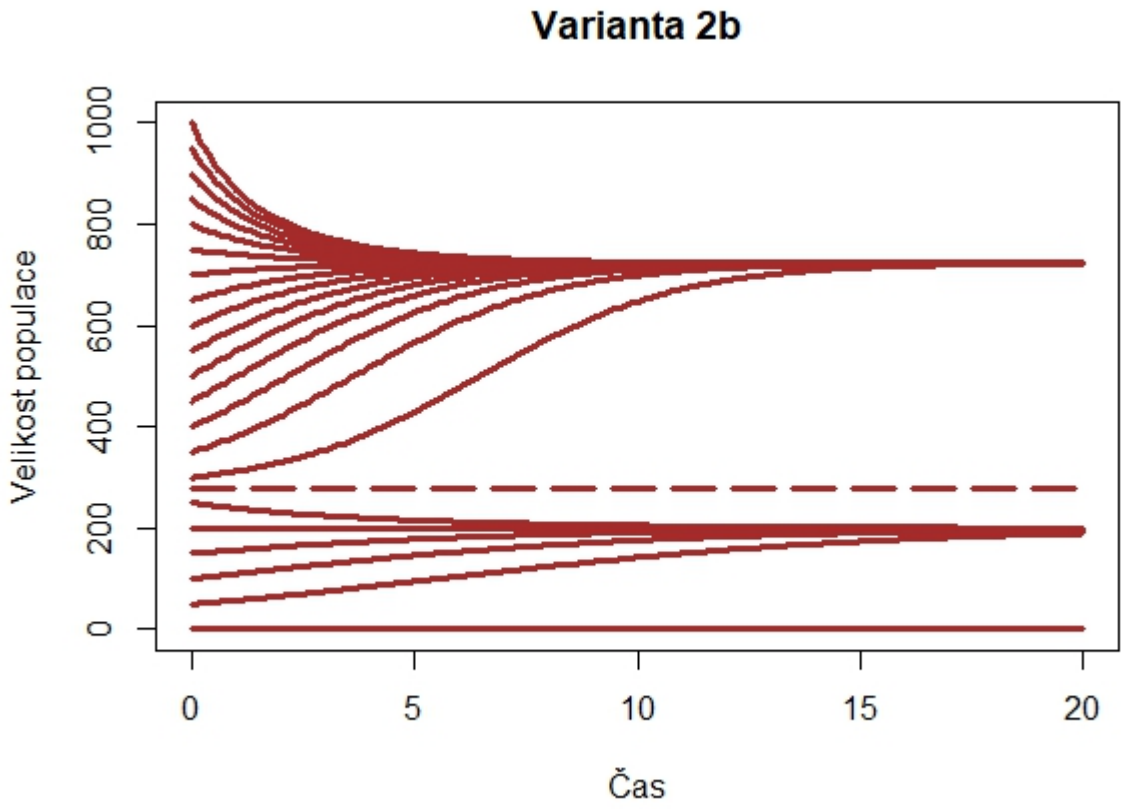
Tuto možnost lze také interpretovat takto: velikost dynamicky stabilizované populace závisí na historii. Pokud populace invadovala do prostředí obsazeného populací predátora, je stabilizovaná populace malá. Naopak, pokud do prostředí se stabilizovanou populací invadovala populace predátora, je stabilizovaná populace velká.

c) $N_{krit} \leq \frac{S}{r}$:
Pokud $N_0 > N_3$, velikost populace se ustálí na rovnovážné hodnotě N_4. Pokud $N_0 < N_3$, ustálí se na hodnotě 0. Pro $N_0 = N_3$ zůstane velikost populace konstantní. Predátor tedy zmenší ustálenou velikost populace nebo populaci vyhubí v závislosti na její počáteční velikosti – obrázek 3.12 pro nastavení parametrů:

$$S = 200,\ \ r = 1,\ \ K = 1000,\ \ N_{krit} = 400,\ \ N_0 \in \{0; 50; \ldots; 1000\}$$
$$\Rightarrow N_1 = 0,\ \ N_3 \doteq 276,\ \ N_4 \doteq 724$$

A obdobným způsobem jako v předchozím případě přidáme k předdefinovaným počátečním počtům také počáteční počet $N_0 = N_3$ a vykreslíme pro něj čárkovanou čáru:

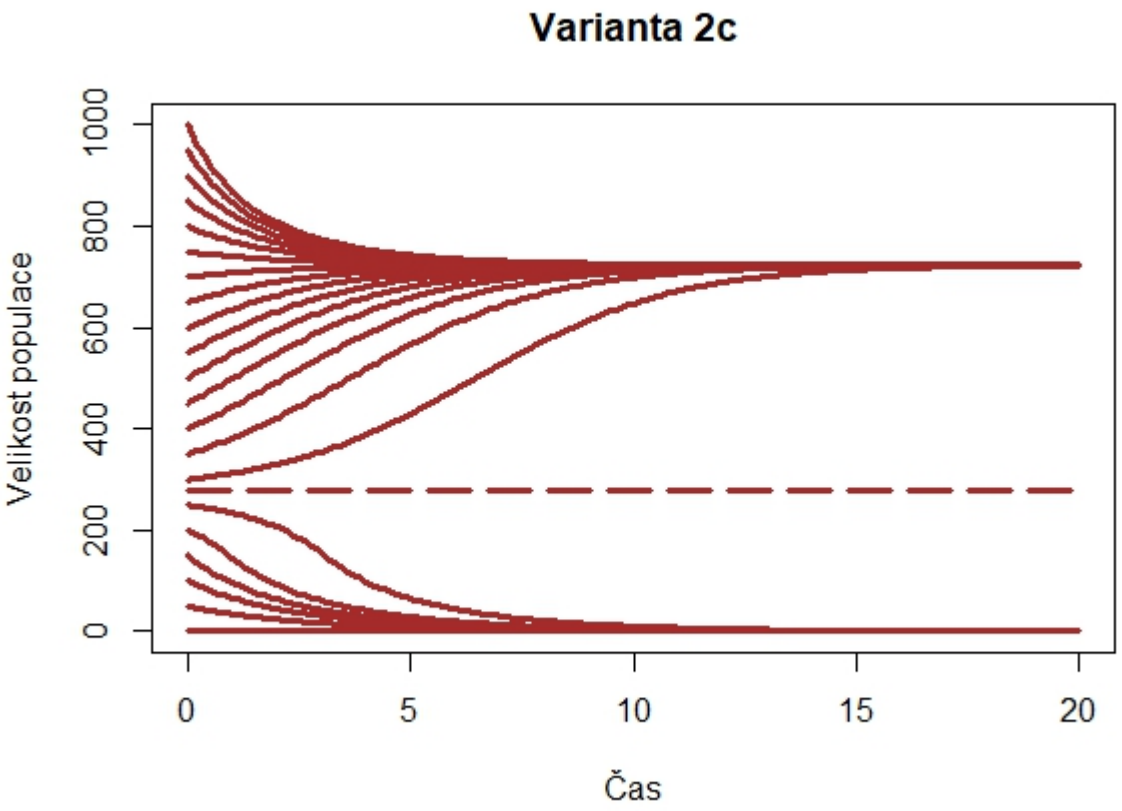
```
# Přidání řešení N3:
podminky<-c("N"=K/2*(1-sqrt(1-4*S/(r*K))))
reseni[,22]<-as.data.frame(ode(podminky,t,soustava,parametry,
                               metoda))$N
lines(t,reseni[,22],lwd=3,lty=5,col="brown")
```



Obrázek 3.11: Varianta 2b

Předešlé možnosti chování modelu platí pro spojitý případ (3.23). V diskrétním případě (3.22) je situace složitější. Obrázek 3.13 a následující kód ilustrují možnost 2a) (tj. $S < \frac{1}{4} \cdot r \cdot K, N_{krit} > N_3$) v diskrétním případě pro $t \in \{0; 0,1; \ldots; 5\}$. V levém grafu jsou vykreslena všechna řešení najednou, v pravém je vybráno jedno řešení pro $N_0 = 1000$. Pro větší přehlednost jsou po sobě následující hodnoty zobrazené jako červené body, propojené tenkou šedou čarou. Můžeme tak vidět, že v těchto případech se velikost populace v čase neustálí na jedné hodnotě, ale osciluje kolem ní. Nulová počáteční velikost však stejně jako ve spojitém případě vede na nulové řešení.

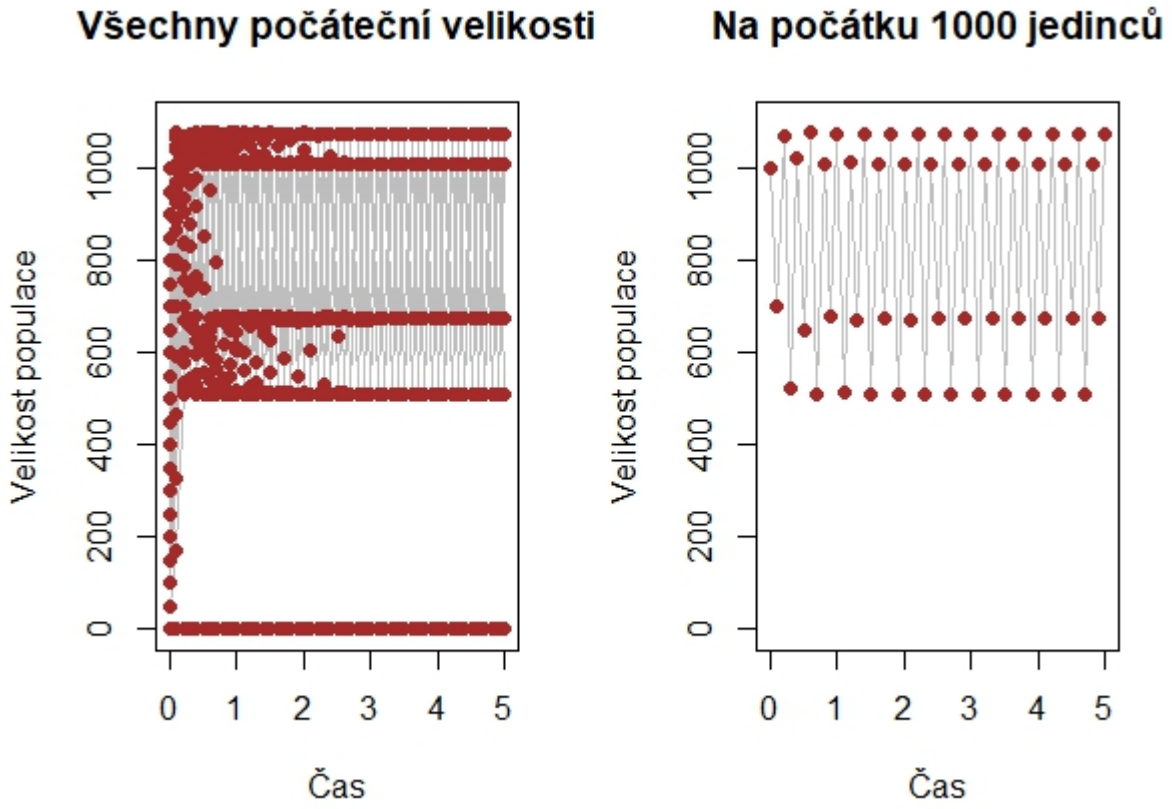
```
r<-3.2
K<-1000
Nkrit<-500
S<-300

t<-seq(0,5,by=0.1)
soustava<-function(t,prom,par) {
  return(list(c(prom["N"]+par["r"]*prom["N"]*(1-prom["N"]/par["K"])-
                  p(prom["N"],par["Nkrit"],par["S"]))))
}
parametry<-c("r"=r,"K"=K,"Nkrit"=Nkrit,"S"=S)
metoda<-"iteration"

reseni<-as.data.frame(matrix(NA,51,21))

for(i in 0:20) {
```



Obrázek 3.12: Varianta 2c

```
  podminky<-c("N"=50*i)

  reseni[,i+1]<-as.data.frame(ode(podminky,t,soustava,parametry,
                                  metoda))$N
}

par(mfrow=c(1,2)) # nastavení dvou grafů vedle sebe

plot(0,
     0,
     cex=0,
     xlim=c(0,5),
     ylim=c(0,1100),
     main="Všechny počáteční velikosti",
     xlab="Čas",
     ylab="Velikost populace")

for(i in 0:20) {
  lines(t,reseni[,i+1],lwd=1,col="gray")
}
for(i in 0:20) {
  points(t,reseni[,i+1],pch=19,col="brown")
}

plot(0,
```

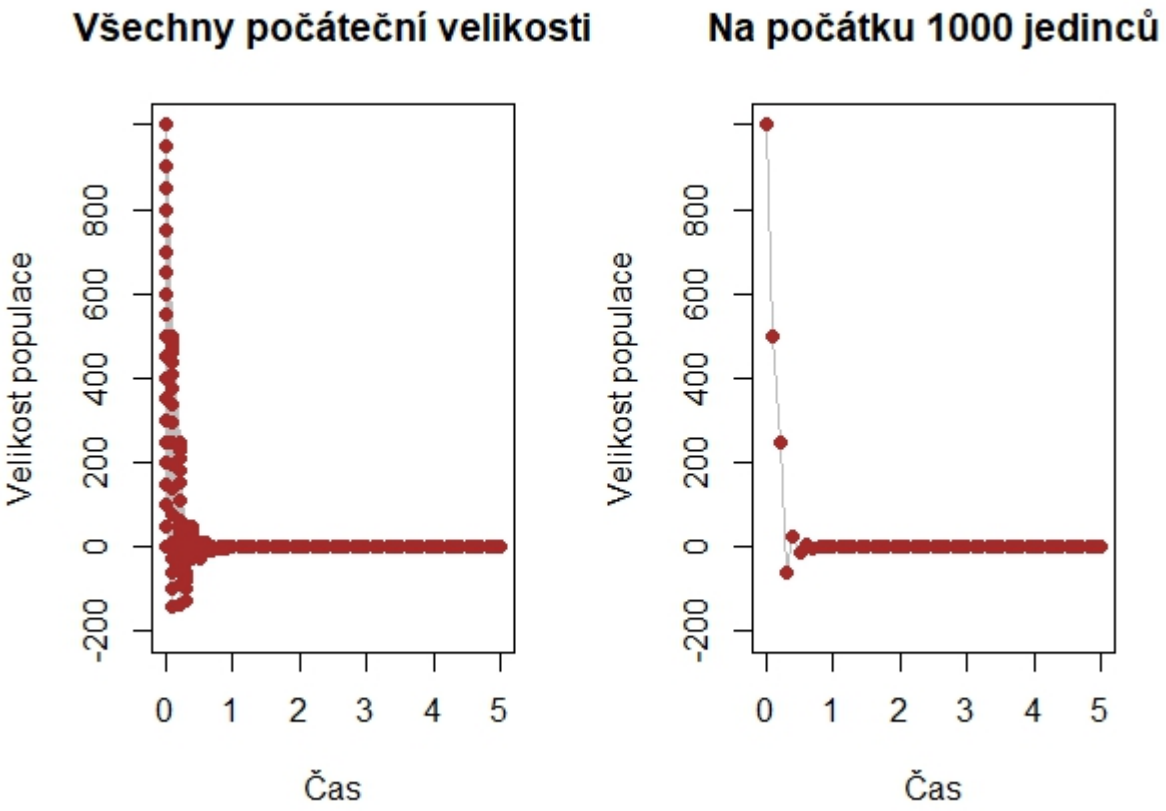
```
      0,
      cex=0,
      xlim=c(0,5),
      ylim=c(0,1100),
      main="Na počátku 1000 jedinců",
      xlab="Čas",
      ylab="Velikost populace")

lines(t,reseni[,21],lwd=1,col="gray")
points(t,reseni[,21],pch=19,col="brown")

par(mfrow=c(1,1)) # nastavení grafu zpět na celou šířku
```



Obrázek 3.13: Řešení modelu růstu populace pod tlakem nespecializovaného predátora v diskrétní podobě případu 2a

Při určitém nastavení parametrů se nám dokonce může stát, že řešení modelu bude dosahovat záporných čísel, jak ilustruje obrázek 3.14 pro možnost 1a (tj. $S \geq \frac{1}{4} \cdot r \cdot K, N_{krit} \leq \frac{S}{r}$). Příčinou je příliš velký časový krok pro dané hodnoty parametrů modelu. Takové řešení můžete vidět na obrázku 3.14, který byl vygenerován stejným kódem jako obrázek 3.13, ovšem, s pozměněným nastavením rozsahu svislé osy `ylim=c(-200,1000)` a hodnotami parametrů:

$$S = 500, \;\; r = 1, \;\; K = 1000, \;\; N_{krit} = 200$$



Obrázek 3.14: Řešení modelu růstu populace pod tlakem nespecializovaného predátora v diskrétní podobě případu 1a

3.3 Interagující populace

Uvažujme obecnou rovnici (společnou pro diskrétní i spojitý model) 3.20 z předešlého modelu vyjadřující růst populace v omezeném prostředí, tj. rovnici

$$N(t+h) = N(t) + r \cdot N(t) \cdot \left(1 - \frac{N(t)}{K}\right) \cdot h, \quad N(0) = N_0 \tag{3.29}$$

a její diskrétní variantu

$$N(t+1) = N(t) + r \cdot N(t) \cdot \left(1 - \frac{N(t)}{K}\right), \quad N(0) = N_0 \tag{3.30}$$

a spojitou variantu

$$N'(t) = r \cdot N(t) \cdot \left(1 - \frac{N(t)}{K}\right), \quad N(0) = N_0. \tag{3.31}$$

Rovnici neomezeného (exponenciálního) růstu populace

$$N(t+h) = N(t) + r \cdot N(t) \cdot h, \tag{3.32}$$

případně její diskrétní variantu

$$N(t+1) = N(t) + r \cdot N(t) \tag{3.33}$$

nebo spojitou variantu

$$N'(t) = r \cdot N(t), \tag{3.34}$$

lze považovat za speciální případ rovnice 3.29, případně 3.30 nebo 3.29, pro $K = \infty$, neboť

$$\lim_{K \to \infty} \frac{N(t)}{K} = 0$$

pro jakoukoliv hodnotu $N(t)$.

Nyní představíme několik modelů dvou, případně tří, interagujících populací. Modely budeme prezentovat v diskrétní i spojité variantě, analýzu jejich řešení však budeme uvádět spolu s příklady v R pouze pro spojitý případ.

3.3.1 Modely dvou interagujících populací

Nyní budeme uvažovat dvě populace na jednom území. Označme $N_1(t)$, resp. $N_2(t)$, velikost první, resp. druhé, populace v čase t. Pokud by na sebe populace vzájemně nepůsobily, vývoj velikosti každé z nich by bylo možné modelovat pomocí rovnice 3.29; o kterou z populací jde, bychom odlišili dolním indexem u všech parametrů, tedy

$$N_1(t+h) = N_1(t) + r_1 \cdot N_1(t) \cdot \left(1 - \frac{N_1(t)}{K_1}\right) \cdot h$$
$$N_2(t+h) = N_2(t) + r_2 \cdot N_2(t) \cdot \left(1 - \frac{N_2(t)}{K_2}\right) \cdot h$$

Vliv velikosti jedné populace na růst druhé se může realizovat buď prostřednictvím prostředí, které obě populace obývají, nebo přímo.

Model, kdy j-tá populace ovlivňuje prostředí, v němž žije i-tá populace

Nechť $i, j \in \{1; 2\}, i \neq j$. Kvalitu prostředí pro i-tou populaci v našem modelu vyjadřuje jediný parametr K_i – úživnost (nosná kapacita) prostředí. Pokud j-tá populace ovlivňuje prostředí, jeho úživnost již nebude konstanta K_i, ale bude záviset na velikosti j-té populace. Konstantu K_i nahradíme funkcí κ_i argumentu $N_j(t)$. Vývoj i-té populace tedy bude popsán rovnicí

$$N_i(t+h) = N_i(t) + r_i \cdot N_i(t) \cdot \left(1 - \frac{N_i(t)}{\kappa_i(N_j(t))}\right) \cdot h. \tag{3.35}$$

Nyní budeme specifikovat funkci κ_i. Ta musí splňovat dva přirozené předpoklady:

- Pokud není j-tá populace přítomná, je úživnost prostředí nezměněna, tedy rovna původní hodnotě K_i. Přesněji $\kappa_i(0) = K_i$.

- Pokud je j-tá populace veliká, změní úživnost prostředí na hodnotu C_{ij}, tedy

$$\lim_{N_j(t)\to\infty} \kappa_i(N_j(t)) = C_{ij}.$$

Jednoduchá funkce, která splňuje obě podmínky, je funkce lomená s lineární funkcí v čitateli i jmenovateli, tedy

$$\kappa_i(N_j(t)) = \frac{K_i + C_{ij}\cdot\gamma_{ij}\cdot N_j(t)}{1+\gamma_{ij}\cdot N_j(t)}. \tag{3.36}$$

V předpisu funkce se objevuje nový parametr γ_{ij}. Abychom lépe pochopili jeho význam, určeme derivaci funkce κ_i podle velikosti j-té populace v bodě $N_j(t) = 0$:

$$\frac{d}{dN_j(t)}\frac{K_i + C_{ij}\cdot\gamma_{ij}\cdot N_j(t)}{1+\gamma_{ij}\cdot N_j(t)} = \frac{C_{ij}\cdot\gamma_{ij}\cdot[1+\gamma_{ij}\cdot N_j(t)] - \gamma_{ij}\cdot[K_i + C_{ij}\cdot\gamma_{ij}\cdot N_j(t)]}{[1+\gamma_{ij}\cdot N_j(t)]^2}$$

po úpravě čitatele pak dostáváme

$$\frac{d}{dN_j(t)}\frac{K_i + C_{ij}\cdot\gamma_{ij}\cdot N_j(t)}{1+\gamma_{ij}\cdot N_j(t)} = \frac{\gamma_{ij}\cdot(C_{ij}-K_i)}{[1+\gamma_{ij}\cdot N_j(t)]^2}$$

po dosazení za $N_j(t) = 0$ je tedy derivace rovna $\gamma_{ij}\cdot(C_{ij}-K_i)$. Nyní vydělíme výsledek rozdílem $C_{ij}-K_i$ (tedy rozdílem úživnosti prostředí pro i-tou populaci mezi krajními hodnotami (tj. nekonečnou a nulovou) velikosti j-té populace) a získáváme hodnotu γ_{ij}. Můžeme proto γ_{ij} chápat jako relativní změnu úživnosti prostředí pro i-tou populaci způsobenou velmi malou (invazní) j-tou populací vzhledem k celkové možné změně úživnosti prostředí i-té populace.

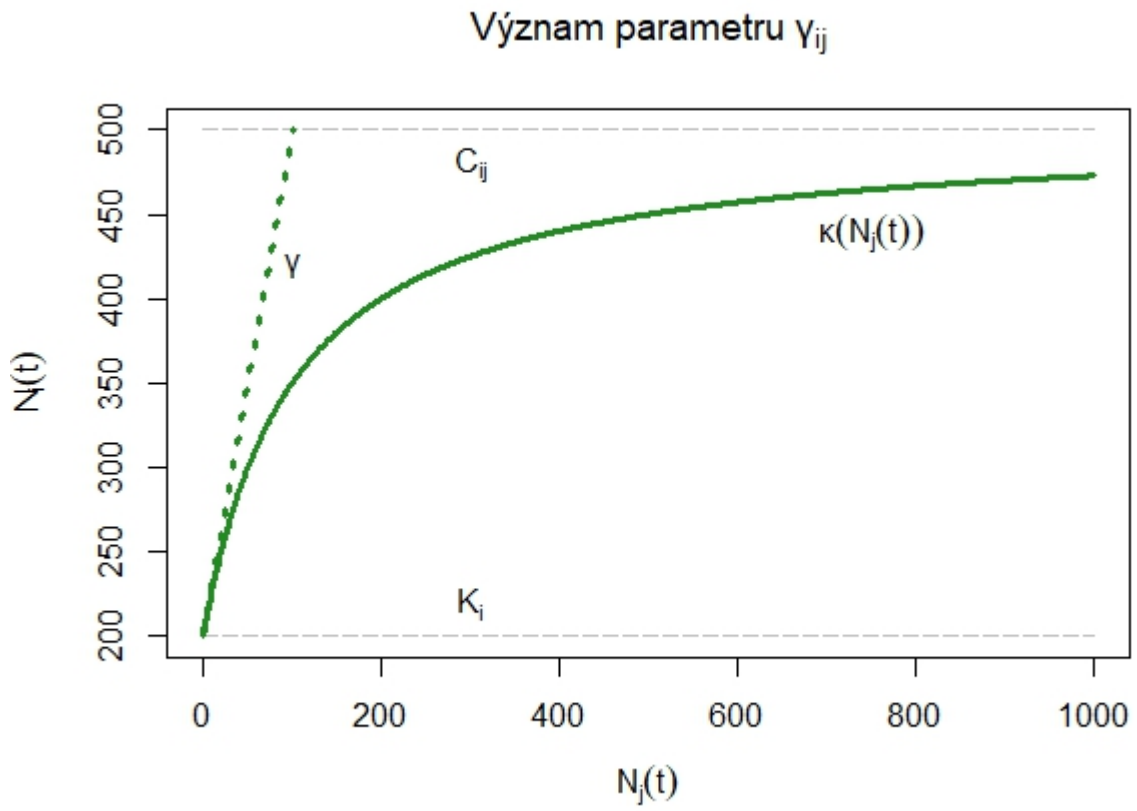
Pro lepší porozumění je situace zobrazena na obrázku 3.15, kde je funkce $\kappa_i(N_j(t))$ vykreslena jako tlustá zelená křivka o hodnotě $K_i = 200$ pro nulovou velikost populace N_j a $C_{ij} = 500$ pro nekonečně velkou populaci N_j (hodnoty 200 a 500 jsou zvýrazněny šedou čárkovanou čarou). Parametr γ_{ij} je znázorněn jako strmost tečny ke křivce v bodě $N_j(t) = 0$ (zelená čárkovaná přímka). Při nastavení použitém pro obrázek, kde $\gamma_{ij} = 0,01$ je z grafu zřejmé, že strmost tečny odpovídá dosažení celého rozdílu $C_{ij} - K_i$ (a tedy ustálení $N_i(t)$ na hodnotě C_{ij} při zvětšení populace $N_j(t)$ z 0 na 100 jedinců. V modelu je však dynamika změny velikosti populace $N_j(t)$ daná předpisem funkce $\kappa(N_j(t))$ pomalejší a křivka se od přímky odklání k nižším hodnotám (roste pomaleji a k C_{ij} konverguje až v nekonečnu).

Obecně je přirozené předpokládat, že konstanty K_i a C_{ij} jsou nezáporné (v prostředí nemůže být populace záporné velikosti) a že alespoň jedna z nich je kladná (prostředí populaci někdy uživí). Vztah konstant K_i a C_{ij} vyjadřuje ekologickou klasifikaci vztahu j-té populace k i-té:

- $K_i = C_{ij}$
  j-tá populace je vůči i-té *neutrální*.

- $K_i > C_{ij}$
  j-tá populace je *amensálem*[22] i-té populace.

[22] *Amensalismus* je populační vztah, při němž jedna populace uvolňuje do prostředí odpadní produkt nebo speciální látku, která populaci jiného druhu ovlivňuje negativně (potlačuje růst a vývoj, způsobí i zánik) [18].



Obrázek 3.15: Význam parametru γ_{ij}

- $K_i < C_{ij}$
  j-tá populace je komensálem[23] populace i-té. V této situaci můžeme dále rozlišit:
  - $Ki = 0$ – i-tá populace by bez přítomnosti j-té nepřežila; j-tá populace je obligátním komensálem populace i-té.
  - $K_i > 0$ – i-tá populace přežívá i bez přítomnosti j-té; j-tá populace je fakultativním komensálem populace i-té.

Z rovností 3.35 a 3.36 dostáváme model vývoje velikostí dvou interagujících populací ve tvaru soustavy rovnic

$$\begin{aligned} N_1(t+h) &= N_1(t) + r_1 \cdot N_1(t) \cdot \left(1 - \frac{N_1(t) \cdot [1 + \gamma_{12} \cdot N_2(t)]}{K_1 + C_{12} \cdot \gamma_{12} \cdot N_2(t)}\right) \cdot h, \\ N_2(t+h) &= N_2(t) + r_2 \cdot N_2(t) \cdot \left(1 - \frac{N_2(t) \cdot [1 + \gamma_{21} \cdot N_1(t)]}{K_2 + C_{21} \cdot \gamma_{21} \cdot N_1(t)}\right) \cdot h. \end{aligned} \tag{3.37}$$

Tuto soustavu rovnic můžeme pro $h \to 0$ konkrétně zapsat jako soustavu rovnic diferenciálních

[23] *Komensalismus* je populační vztah, při němž jedna populace využívá jinou bez jejího poškozování (jedna populace má ze vztahu prospěch, druhá není ovlivněna) [18].

$$
\begin{aligned}
N_1'(t) &= r_1 \cdot N_1(t) \cdot \left(1 - \frac{N_1(t) \cdot [1 + \gamma_{12} \cdot N_2(t)]}{K_1 + C_{12} \cdot \gamma_{12} \cdot N_2(t)}\right), \\
N_2'(t) &= r_2 \cdot N_2(t) \cdot \left(1 - \frac{N_2(t) \cdot [1 + \gamma_{21} \cdot N_1(t)]}{K_2 + C_{21} \cdot \gamma_{21} \cdot N_1(t)}\right),
\end{aligned}
\tag{3.38}
$$

nebo pro $h = 1$ jako soustavu rovnic diferenčních

$$
\begin{aligned}
N_1(t+1) &= N_1(t) + r_1 \cdot N_1(t) \cdot \left(1 - \frac{N_1(t) \cdot [1 + \gamma_{12} \cdot N_2(t)]}{K_1 + C_{12} \cdot \gamma_{12} \cdot N_2(t)}\right), \\
N_2(t+1) &= N_2(t) + r_2 \cdot N_2(t) \cdot \left(1 - \frac{N_2(t) \cdot [1 + \gamma_{21} \cdot N_1(t)]}{K_2 + C_{21} \cdot \gamma_{21} \cdot N_1(t)}\right).
\end{aligned}
\tag{3.39}
$$

V případě komensalismu může dojít i k tomu, že populace komensála při rostoucí velikosti zvětšuje úživnost prostředí nade všechny meze, $lim_{N_j(t)\to\infty}\kappa_i(N_j(t)) = \infty$. Nejjednodušší funkce, která modeluje tento jev, je funkce lineární

$$
\kappa_i(N_j(t)) = K_i + \gamma_{ij} \cdot N_j(t).
$$

Zde kladný parametr γ_{ij} vyjadřuje absolutní nárůst úživnosti prostředí pro i-tou populaci způsobený j-tou populací o jednotkové velikosti.

Model konkurence (kompetice)

Model konkurence vyjadřuje interakci mezi populacemi způsobenou sdílenými požadavky na zdroj, který je omezeně dostupný [20]. Obě populace si tímto vzájemně snižují úživnost prostředí. Velikosti obou populací se ustálí na nových rovnovážných hodnotách menších, než byly hodnoty pro izolované (vzájemně se neovlivňující) populace.

Literatura

[1] Adam, J.A.: Mathematics in Nature. Princeton and Oxford (2003)

[2] Barnes, B., Fulford, G.R.: Mathematical Modelling With Case Studies: A Differential Equation Approach Using Maple. CRC Press, London (2002)

[3] Begon, M., Harper, J.L., Townsend, C.R.: Ecology: individuals, populations and communities. Blackwell Scientific Publications, Oxford (1990)

[4] Ellner, S.P., Guckenheimer, J.: Dynamic Models in Biology. Princeton University Press, Princeton, New Jersey (2006)

[5] Elton, C.: The Ecology of Invasion by Animals and Plants. Methuen, London (1958)

[6] Eriksson, O.: Sensitivity and Uncertainty Analysis Methods, with Applications to a Road Traffic Emission Model. Dissertation thesis, Department of Mathematics, Linköping University (2007)

[7] Gander, W., Hřebíček, J.: Solving Problems in Scientific Computing Using Maple and MATLAB. 4th, expanded and rev. ed. Springer, Heidelberg (2004)

[8] Gause, G.F.: The struggle for existence. Williams and Wilkins, Baltimore (1934)

[9] Holčík, J., Fojt, O.: Modelování biologických systémů (Vybrané kapitoly). ÚBMI FEI VUT, Brno (2001)

[10] Holling, C.S.: The functional response of predator to prey density and its role in mimicry and population regulation. Mem. Entomol. Soc. Canada, vol. 45, pp. 1-60 (1965)

[11] Hřebíček, J., Hejč, M., Holoubek, I., Pešl, J.: Current Trends in Environmental Modelling with Uncertainties. In Proceedings of the iEMSs Third Biennial Meeting "Summit on Environmental Modelling & Software". Burlington : International Environmental Modelling and Software Society, pp. 342-347 (2006)

[12] Hřebíček, J., Žižka, J.: Vědecké výpočty v biologii a biomedicíně [online]. [cit 2010-07-29]. Dostupný z WWW: <`http://portal.med.muni.cz/download.php?fid=502`>

[13] Hřebíček, J., Škrdla, M.: Úvod do matematického modelování. Masarykova univerzita, Brno (2006)

[14] Isukapalli, S.: Uncertainty analysis of transport transformation models. Ph.D. thesis, State University of New Jersey, New Brunswick, New Jersey (1999)

[15] Kalas, J., Pospíšil, Z.: Spojité modely v biologii. Masarykova univerzita, Brno (2001)

[16] Kalas, J., Ráb, M.: Obyčejné diferenciální rovnice. Masarykova univerzita, Brno (2001)

[17] Kobza, A.: Diferenční rovnice ve středoškolské matematice. Diplomová práce, Masarykova univerzita, Brno (2001)

[18] Kulhavy, J.: Biotické interakce [online]. [cit 2010-07-29]. Dostupný z WWW: <http://www.uel.cz/download/Prednaska_6_Bioticke%20interakce.pdf>

[19] Lepš, J.: Růst populace [online]. [cit 2010-07-29]. Dostupný z WWW: <http://botanika.bf.jcu.cz/suspa/vyuka/materialy/Rustpopulace.ppt>

[20] Lepš, J.: Kompetice [online]. [cit 2010-07-29]. Dostupný z WWW: <http://botanika.bf.jcu.cz/suspa/vyuka/materialy/Kompeticeprvacka.ppt>

[21] Leslie, P.H.: Some further notes on the use of matrices in population mathematicics. Biometrika, vol. 35, pp. 213-245 (1948)

[22] MacArthur, R.H.: Fluctuations of animal populations and a measure of community stability. Ecology, vol. 36, pp. 533–536 (1955)

[23] Madzia, L.: Zajíc polní [online]. [cit 2010-07-29]. Dostupný z WWW: <http://www.prirodainfo.cz/karta.php?cislo=3080.00
textgreater

[24] Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Roučka, Š., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., Scopatz, A.: SymPy: symbolic computing in Python. The open access journal for computer science 3:e103 (2017) doi:10.7717/peerj-cs.103

[25] Novák, V.: Základy fuzzy modelování. BEN – technická literatura, Praha (2000)

[26] Odum, E.P.: Fundamentals of Ecology. W.B. Saunders Company, Philadelphia (1971). Český překlad: Základy ekologie. Academia, Praha (1977)

[27] Pešl, J.: Uncertainty handling in the environmental modeling using computer algebra systems with online data manipulation. Ph.D. thesis, Masaryk University, Brno (2005)

[28] Pospíšil, Z.: Dynamika populací s oddělenými generacemi [online]. [cit 2010-07-29]. Dostupný z WWW: <http://www.math.muni.cz/~pospisil/FILES/DPNG.pdf>

[29] Saltelli, A., Chan, K., Scott, E.M.: Sensitivity Analysis. John Wiley and Sons, Ltd.: West Sussex, England (2000)

[30] Saltelli, A.: Global Sensitivity Analysis: An Introduction. In Proc. 4th International Conference on Sensitivity Analysis of Model Output, pp. 27 – 43 (2004)

[31] Sharov, A.: Quantitative Population Ecology. On-Line Lectures [online]. [cit 2010-07-29]. Dostupný z WWW: <http://home.comcast.net/~sharov/PopEcol/>

[32] Soetaert, K., Petzoldt, T., Setzer, R. W. Solving Differential Equations in R: Package deSolve. Journal of Statistical Software, 33(9), 1–25 (2010) doi:10.18637/jss.v033.i09.

[33] The Council for Regulatory Environmental Modeling.: Draft Guidance on the Development, Evaluation, and Application of Regulatory Environmental Models [online]. [cit 2010-07-29]. Dostupný z WWW: <http://www.modeling.uga.edu/tauc/other_papers/CREM%20Guidance%20Draft%2012_03.pdf>

[34] Ushey, K., Allaire, J., Tang, Y.: reticulate: Interface to 'Python' (2023). Dostupný z WWW: <https://rstudio.github.io/reticulate>