Mendel University in Brno **2022**

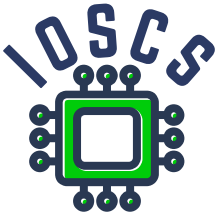
1st International Conference on Open Source tools in Computer Science university education

Conference Proceedings

Jiří Rybička (Ed.)
Mendel University in Brno

October 25–26, 2021, building Z, room Z15

Project: Innovative Open Source Courses for Computer Science Curriculum



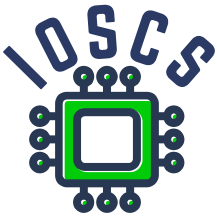
25.–26. 10. 2021

Co-funded by the Erasmus+ Programme of the European Union
West Pomeranian University of Technology Szczecin
UNIVERSITY OF ŽILINA
Mendel University in Brno

Reviewers: prof. RNDr. Vojtech Bálint, CSc., RNDr. Michal Kaukič, CSc.,
doc. Ing. Miroslav Kvaššay, PhD., University of Žilina
Project: Innovative Open Source Courses for Computer Science Curriculum

ISBN 978-80-7509-898-6 (online; pdf)
DOI https://doi.org/10.11118/978-80-7509-898-6

Project: Innovative Open Source Courses for Computer Science Curriculum



This material was created as one of the activity of the project “Innovative Open Source Courses for Computer Science Curriculum”, funded by the Erasmus+ grant no. 2019-1-PL01-KA203-065564.

Project information

Project was implemented under the Erasmus+.
Project name: **“Innovative Open Source courses for Computer Science curriculum”**
Project nr: **2019-1-PL01-KA203-065564**
Key Action: **KA2 – Cooperation for innovation and the exchange of good practices**
Action Type: **KA203 – Strategic Partnerships for higher education**

Consortium
ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY W SZCZECINIE
MENDELOVA UNIVERZITA V BRNĚ
ŽILINSKÁ UNIVERZITA V ŽILINĚ

Erasmus+ Disclaimer
This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the
Erasmus+ Programme
of the European Union

Obsah

Foreword

Jiří Rybička

Multiplier events – conferences are one of the main activities of the project "Innovative Open Source Courses for Computer Science Curriculum". The first of the two planned conferences took place on October 25 and 26, 2021 in room Z15 of building Z at Mendel University in Brno. The working languages of the conference were English, Czech and Slovak.

A "1st International Conference on Open Source tools in Computer Science university education" included papers focused on project information, but also a section of papers devoted to more general aspects of open source software – analysis with Jupyter (dr. Mařík) and the use of R system in statistical evaluations and their interpretation (dr. Vencálek – chairman of the Czech statistical society).

Due to major restrictions in the ongoing Covid-19 pandemic, as many speakers and listeners as would have been appropriate could not participate in the conference. Nevertheless, it was possible to reach the number of 50 participants and give the conference the necessary weight.

The conference was divided into two days. On the first day, contributions related to general information about the project and contributions from the extended context of open source programs were heard, while on the second day contributions focused on clarifying the concepts of individual courses that are to be created within the project.

This proceedings sumarizes all contributions on the conference. The main section contains the contributions of the first day, the second section called "Courses and their concepts" concentrates the contributions presented on the second day.

We have attached several photos to remind you of the atmosphere of the conference.

DOI https://doi.org/10.11118/978-80-7509-898-6-0005

Fig. 1: *A contribution of dr. Remigiusz Olejnik*

Fig. 2: *A reflection of Summer School by J. Diviš*

Fig. 3: *A contribution of dr. Robert Mařík*

Fig. 4: *Conference audience*

Hlavní sekce

Main section

Innovative Open Source courses for Computer Science curriculum — Erasmus+ KA2 Strategic Partnership Project

Remigiusz Olejnik

1 Introduction

This proceedings is a result of the *1st International Conference on Open Source tools in Computer Science university education* held in Brno on 25–26th of October 2021. The conference is first multiplier event planned as part of the project "Innovative Open Source courses for Computer Science curriculum".

2 Information about the project

The project "Innovative Open Source courses for Computer Science curriculum (IOSCS)" is funded under EU Erasmus+ Programme in Key-Action 2 "Cooperation for innovation and the exchange of good practices", Action Type "Strategic Partnerships for higher education". Project duration is from 1st of September 2019 until 31st of August 2022.

2.1 Project participants and teams' leaders

Project consortium is composed of three Higher Education Institutions in Central Europe:

1. Zachodniopomorski Uniwersytet Technologiczny w Szczecinie — ZUT (PL) [coordinator]
2. Mendelova univerzita v Brně — MENDELU (CZ)
3. Žilinská univerzita v Žiline — UNIZA (SK)

The teams at each participating university are lead by:

1. ZUT — dr. inż. Remigiusz Olejnik
2. MENDELU — doc. Ing. Jiří Rybička, Ph.D.
3. UNIZA — RNDr. Aleš Kozubík, PhD.

DOI: https://doi.org/10.11118/978-80-7509-898-6-0009

2.2 Project goals

Main goal of the project is **to develop 6 study courses, based on Open Source approach**. The courses will cover fields of programming, mathematics, operating systems, embedded systems, systems controlled by computer and other hardware and/or software applications.

The results of IOSCS project will be:

- descriptions of the courses
- materials necessary for lectures and practical classes
- handbook composed of the materials necessary to follow the courses by the students

The courses will be tested during summer intensive courses — the university teachers from the consortium organizations will deliver the courses to the selected group of the students. All the results will be publicly available on the website of the project IOSCS and involved organizations. Moreover they will be spread during International Conferences on Open Source tools in Computer Science university education

2.3 Selected courses

The consortium decided to develop a materials for the following courses:

1. Open Source tools for text processing
2. Probability and statistics with programming in R
3. Mathematical Analysis supported by wxMaxima
4. Programming language Lua
5. Mobile Application Development
6. Wireless Signal Processing in GNU Radio Environment

2.4 Intellectual Outputs

In order to develop the courses the works are divided into three phases:

1. IO1 — Developing courses based on Open Source tools **[already finished]**
2. IO2 — Developing the materials for the intensive courses **[already finished]**
3. IO3 — Developing a handbook on Open Source tools for Computer Science study programmes implemented at university level

2.5 Summer intensive courses

The courses are to be tested during summer intensive courses. The project plans two such events:

1. 2021 — Žilina
2. 2022 — Brno

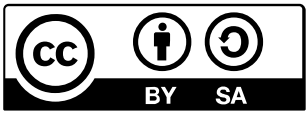
3 Multiplier events

The results of the project are to be spread during two international conferences:

1. 2021 — Brno: 1st International Conference on Open Source tools in Computer Science university education
2. 2022 — Žilina: 2nd International Conference on Open Source tools in Computer Science university education

Author

dr. inż. Remigiusz Olejnik, Zachodniopomorski Uniwersytet Technologiczny w Szczecinie



Jupyter, aneb vědecké výpočty moderně a snadno

Robert Mařík

Rychlý rozvoj Internetu, virtualizačních nástrojů a cloudových služeb v posledních letech zcela změnil způsob práce uživatelů, kteří se zabývají vědeckými či jakýmikoliv výpočty. Výpočetní nástroje mají běžně možnost provádět výpočty v cloudu a uživatel může tedy pracovat v podstatě na jakkoli výpočetně slabém zařízení. Samozřejmostí je sdílení souborů se spolupracovníky nebo se všemi uživateli Internetu. Pokud není přímo k dispozici možnost používat některý z verzovacích systémů (například kvůli binárnímu ukládání dat), mívají výpočetní prostředí v sobě integrovánu možnost ukládání verzí, návratu k dřívější verzi nebo zobrazení rozdílu mezi verzemi. Samozřejmostí je, že delší poznámky se nepíší mezi řádky kódu za komentářové znaky, ale je možné přirozeně kombinovat výpočty s textem. Mimo jiné i díky tomu je moderní věda reprodukovatelná, otevřená a férová (z ustálených anglických obratů *reproducible science*, *open science* a *fair science*).

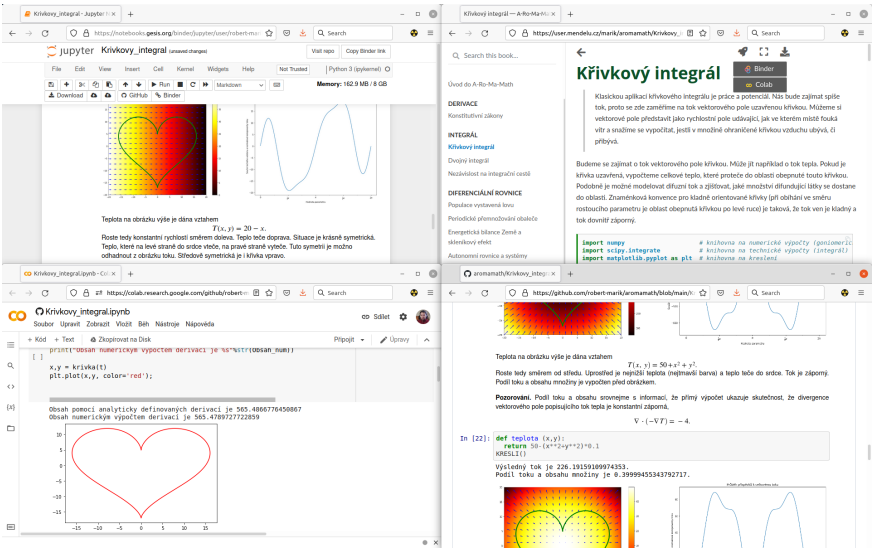
1 Jupyter

V tomto příspěvku si představíme otevřený nástroj, který umožní provádět libovolné výpočty moderním přístupem se všemi vlastnostmi popsanými v úvodu, tj. výpočty v cloudu, s možností sdílení a se správou verzí. Jedná se prostředí Jupyter, resp. Jupyter Notebook, [1]. Jupyter je ekosystém vědeckých nástrojů pro numerické výpočty, symbolické výpočty, kreslení grafů a obrázků, statistické zpracování dat, analýzu obrazových nebo zvukových dat, analýzu textu, strojové učení a rozpoznávání textu či obrázků a mnoho dalšího.

Z uživatelského hlediska se Jupyter chová jako webová aplikace pro přístup k programovacím jazykům Python, Julia a statistickému jazyku R. Odsud, z názvů těchto tří jazyků, je ostatně složen i název projektu. Zejména volba jazyka Python se ukazuje velice šťastná, protože ve všech oborech zmíněných v předchozím odstavci, patří Python a jeho knihovny mezi nejvýkonnější nástroje. To platí ať výkon posuzujeme z hlediska vzdělávání budoucích inženýrů a výpočtářů, nebo z hlediska výpočtářů řešících úlohy v inženýrské praxi. To, že vše je umístěno v jednotném prostředí, umožňuje zmíněné nástroje použít jako stavební kostky k řešení sofistikovaných úloh, které mohou kombinovat náročnou práci s obrazovými vstupy, numerické výpočty nad obrovskými

 DOI: https://doi.org/10.11118/978-80-7509-898-6-0012

objemy dat, vizualizaci výstupů a případně export výpočtů ve formě csv souborů nebo v jakémkoliv běžně užívaném formátu.[1]



Obr. 1: *Jupyter zápisník otevřený ve čtyřech prostředích. Vpravo dvě statické verze pro vystavení nebo sdílení, vlevo dvě dynamické verze umožňující spouštět vlastní příkazy.*

Bylo by faux-paux mluvit o moderních výpočetních metodách a rozepisovat se detailně o způsobu práce, příkazech, nastaveních. Všechno máte na dosah ruky, tak proč to rovnou nezkusit? Ukázkou jednoduchých učebních textů opírajících se o výpočty v prostředí Jupyter je webový miniportál A-Ro-Ma-Math [2]. Jedná se o takzvaný Jupyter–book, kniha sestavená se zápisníků Jupyter–notebook. Jeden z těchto zápisníků, zápisník Křivkový integrál, je na obrázku otevřen ve čtyřech různých provedeních ze čtyřech různých lokalit a tím ve čtyřech různých podobách.

- Vpravo nahoře je dokument otevřen ze základního umístění na domovské instituci autora. Po rozkliknutí ikonky s raketou v pravém horní rohu se nabízí možnost otevřít dokument v interaktivní pobobě s možností dokument měnit a spouštět příkazy. V současnosti jsou aktivní možnosti využít projektů Binder a Colab.
- V levé polovině je dokument otevřen v prostředí Binder (nahoře) a Colab (dole). Zde pracujete s kopií původního souboru a můžete si jej libovolně upravovat, expe-

[1] Z inženýrského hlediska zde chybí pravděpodobně jenom metoda konečných prvků, pro kterou sice také máme k dispozici knihovnu, ale ta nedosahuje kvalit svých komerčních konkurentů.

rimentovat s příkazy a poté uložit buď do lokálního souboru, nebo v případě Colab i na Google Drive nebo do repozitáře GitHub.

- Vpravo dole je tentýž dokument v prostředí Github, v prostředí zaměřeném na správu verzí dokumentů a softwarových projektů. Toto prostředí umí přímo renderovat Jupyter zápisníky tak, jak je vidíme při práci. Používá se pro sdílení, ale sdílet je možno samozřejmě i soubor uložený v Colab na Google Drive.

Díky balíčkovacímu systémy jazyka Python je možné a snadné si Jupyter nainstalovat na vlastní počítač s libovolým běžným operačním systémem. Potom je možno pracovat bez připojení k Internetu a být tak odolní vůči případným omezením služeb. Práce v cloudu jako je Binder či Colab je však často pohodlnější. Odpadá například veškeré ukládání při přenášení na jiný počítač (učebna, vedlejší kancelář, práce, domov, ...). Přes nutnost přistupovat ke cloudovým službám není nutné se bát malého výkonu. Naopak. Pro strojové učení je začátečníkům prostředí Colab zpravidla doporučováno, protože poskytne krátkodobě dostatečný výpočetní výkon i uživatelům bez drahých grafických karet, které jsou pro tyto aplikace zpravidla téměř nutností.

2 Knihovny pro Jupyter

Pokud tedy máte spuštěn Jupyter v prostředí Colab, Binder, nebo z URL `https://jupyter.org/try` nebo jste investovali úsilí do instalace na vlastní počítač, můžete se pustit do učení a do práce. Níže si uvedeme jednotlivé součásti. Každá z nich má obsáhlou dokumentaci a nápovědu, často i podrobnou nástěnku a ukázkami výstupů. Proto je zbytek textu míněn spíše jako návěstí u rozcestníku. Co čtenáře zaujme, to může okamžitě vyzkoušet a není těžké vygooglit jednoduché i sofistikované ukázky možností a aplikací.

2.1 Python

Python je jeden z nejjednodušších programovacích nebo skriptovacích jazyků. Pokud nějaký skriptovací jazyk používáte, nemusíte se pro začátek Python pravděpodobně vůbec učit. Stačí se na příkladech seznámit se syntaxí. Můžete se rovnou podívat na použití v dokumentu [3], který oproti svému názvu není zaměřen pouze na knihovnu Scipy, ale pokrývá i Python obecně a řadu dalších knihoven, zmíněných níže.

Pokud jste v programování začátečníkem, existuje řada kurzů, jak do problematiky jazyka Python snadno proniknout, například [4] nebo v češtině [5].

2.2 Numpy

Numerické výpočty zajišťuje knihovna numpy, což je zkratka pro Numerical Python. Knihovna definuje datové struktury a nástroje pro snadnou manipulaci a matematické numerické výpočty s čísly nebo s číselnými poli. S knihovnou je možné se seznámit v tutoriálu [4].

2.3 Scipy

Knihovna pro vědecké výpočty, obsahující nástroje a funkce pro optimalizaci, interpolaci, numerickou integraci, řešení diferenciálních rovnic, grafové algoritmy a další časté inženýrské úlohy. Viz [3], [4]. Knihovna umožňuje ukládání a načítání datových struktur uložených programem Matlab.

V [2] je knihovna scipy použita k řešení diferenciálních rovnic.

2.4 Matplotlib

Knihovna pro kreslení grafů všeho druhu je knihovna matplotlib. Nejlepší vypovídací hodnotu o možnostech má návštěva domovské stránky projektu [6] a sekce Examples a Tutorials. Další tutoriál je součástí [4].

Nejedná se o jedinou knihovnu pro kreslení grafů v jazyce Python. Pro kreslení statistických grafů je možno zvážit jako alternativu Seaborn, pro kreslení interaktivních grafů umožňujících zoom, posun nebo vypínání některých křivek je Bokeh.

2.5 Pandas

Pandas je knihovna pro práci s datovými soubory, pro práci se seznamy a tabulkami čísel nebo textových údajů. Takové tabulky používáme v databázích nebo v tabulkových procesorech. Přirozeně jsou takové tabulky používány Excelem a co je možné vypočítat v Excelu je možné vypočítat knihovnou Pandas. Navíc můžete díky skriptování snadno automatizovat činnosti, práce je mnohem přehlednější, nejste omezeni velikostí datových souborů snadno se kombinují data z více zdrojů a spousta dalších činností je v Pandas mnohem efektivnější. Proto se vyplatí do studia investovat čas. Překousnout, že první tabulku se součty a průměry budete v Pandas dělat možná dvacetkrát déle než v Excelu. Jakmile stylu práce přijde člověk na kobylku, je práce u běžných činností minimálně stejně efektivní a navíc reprodukovatelná, transparentní (použití příkazů versus postupné klikání v GUI) a snadno začlenitelná do komplexnějších projektů. Bonusem je, že práce s malými datovými soubory stejně náročná jako práce s těmi obrovskými, které byste do Excelu ani nenačetli. Na nástroje typu MS Excel je dobré nezapomínat. Jsou

široce rozšířené, jsou v povědomí mnoha lidí a jsou dobré na krátké analýzy. Na hlubší práci spojenou kromě analýzy například i s čištěním dat je však rozhodně lepší používat Pandas.

Pandas je možné se naučit ve většině kurzů věnovaných využití jazyka Python. Můžete začít například na tutoriálu [4] a pokračovat některým z široké nabídky online kurzů na zpracování dat a práci s daty na portálu [7].

2.6 SymPy

SymPy je kompletní systém počítačové algebry v jazyce Python. Obsahuje diferenciální a integrální počet, diferenciální rovnice, lineární algebru, řešení rovnic, kombinatoriku a další nástroje pro symbolické výpočty. Dokumentace na domovské stránce projektu obsahuje i tutoriál a ukázky je možno spouštět přímo z textu dokumentace.

2.7 Další knihovny

Výše uvedený seznam není jistě vyčerpávající. Slouží pro rychlé seznámení se s těmi nejpodstatnějšími knihovnami. Existuje i celá řada dalších, které jsou specializovány na jiné činnosti. Například scikit-learn nebo TensorFlow pro strojové učení, Biopython s nástroji pro bioinformatiku, PySB pro systémovou biologii a modelování biochemických systémů a řada dalších.

3 Závěr

Výše představené nástroje nejsou interaktivní, jedná se o programovací (skriptovací) jazyk a na rozdíl od nástrojů s GUI je nutné se práci nějakou dobu učit. Nabízí se otázka, za se jedná o skutečně použitelné a používané nástroje, do kterých je vhodné investovat čas a úsilí. Odpověď je možné ilustrovat situací na pravděpodobně nejkvalitnějším a nejobsáhlejším elarningovém portále edX [7]. Zde je naprostá většina kurzů zaměřených na zpracování dat nebo na práci s daty postavena okolo jazyka Python a často v prostředí Jupyter. Nejedná se přitom jenom o univerzitní kurzy, ale i o kurzy připravené technologickými firmami jako IBM. To ukazuje popularitu a garantuje životaschopnost inženýrských výpočtů a zpracování dat jazykem Python v prostředí Jupyter.

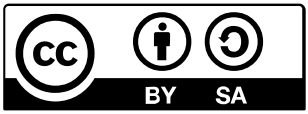
Jako perličku na závěr, která ilustruje kvalitu představených produktů, bych uvedl jeden příklad. Lidé se často ptají, zda je lepší Pandas nebo R. Málokoho napadne do srovnání přibrat ještě dalšího kandidáta. Přirozeně srovnávají dva nejvyspělejší produkty v oblasti statistiky, dva opensource projekty které daleko utekly konkurenci. Oba tyto produkty je možno provozovat v zápisníku Jupyter.

Literatura

1. *Project Jupyter*, https://jupyter.org/
2. Maříк, R. *A-Ro-Ma-Math*, https://user.mendelu.cz/marik/aromamath (web), https://github.com/robert-marik/aromamath/ (GitHub repozitář)
3. *Scipy Lecture notes*, https://scipy-lectures.org/
4. *Python Tutorial*, https://www.w3schools.com/python
5. *Nauč se Python*, https://naucse.python.cz/
6. *Knihovna matplotlib*, https://matplotlib.org/
7. *edX*, https://www.edx.org/

Autor

Doc. Mgr. Robert Mařík, Ph.D., Ústav matematiky, Lesnická a dřevařská fakulta, Mendelova univerzita v Brně, Zemědělská 1, 613 00 Brno, e-mail: robert.marik@mendelu.cz



Pohled účastníka na Letní školu

Jan Diviš

Na konferenci o inovativních Open Source nástrojích bych chtěl přispět příspěvkem o letní škole v Žilině. Jednalo se o studijní pobyt v zahraničí v rámci projektu Erasmus+. Byl to první ročník tohoto mezinárodního projektu. Studenti ze Štětína, Brna a Žiliny byli součástí kurzů, a to buď osobně nebo distančně. My jsme si vybrali prezenční formu.

Projekt byl jednotýdenní, časová náročnost nebyla příliš dramatická a mohli jsme si vybrat až ze šesti kurzů, z nichž jsme si dále zvolili dva, které budeme navštěvovat. Byly to kurzy týkající se problematiky informačních technologií. Já sám jsem navštěvoval kurzy programování v jazyce Lua a bezdrátové zpracování signálu v radiovém prostředí GNU. Vše bylo plně hrazeno Evropskou unií.

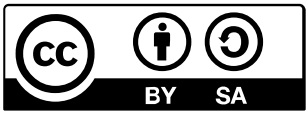
Zprvu mě velice překvapila časová náročnost. Docházeli jsme jeden týden, vždy od deváté hodiny ranní do páté hodiny odpolední na tamější univerzitě Fakulty riadenia a informatiky. Mezitím jsme měli dvouhodinové pauzy, aby se prostřídaly všechny skupiny a ostatní kurzy, bylo to nejspíš způsobeno nepříznivou epidemiologickou situací. Tento čas by se dal jistě strávit i lepším způsobem, ovšem okolnosti k tomu nebyly nakloněny. Vše bylo v angličtině, vyučující byli vždy zástupci univerzit studentů, kteří se projektu zúčastnili. Bohužel kurzy, které byly vyučovány univerzitou Mendelu, musely být distančně. Panu docentu Rybičkovi i doktoru Hálovi patří ovšem velký obdiv, jak se k celé situaci postavili. Vše proběhlo, jak mělo a dozvěděli jsme se v celku dost zajímavé věci. Další ročník tohoto projektu proběhne u nás v Brně. Doufám, že budu mít tu čest se projektu znovu zúčastnit.

Projekt se mi velice zalíbil, bylo to vůbec poprvé, kdy jsem se mohl něčeho takového zúčastnit. Byla to velice dobrá zkušenost, sám jsem se vždy něčeho takového ostýchal, ovšem nyní mi to změnilo pohled na věc. Student si zde může nejenom rozšířit obzory na poli vědy, v našem případě informatiky, ale hlavně překonat sám sebe. Všem studentům, kteří mají obavy z něčeho takového, stejně jako jsem měl kdysi já, bych doporučil, aby dali ostych stranou a něco podobného vyzkoušeli taktéž, bylo to skvělé. Student získá hodně nových informací, životních zkušeností a se svými přáteli stráví okamžiky, na které bude vzpomínat do konce života.

DOI: https://doi.org/10.11118/978-80-7509-898-6-0018

Autor

Jan Diviš, Provozně ekonomická fakulta, Mendelova univerzita v Brně, Zemědělská 1, 613 00 Brno, e-mail: jan.divis@mendelu.cz



Daty podložené omyly

Ondřej Vencálek

1 Úvod – analýza dat v našich životech

V březnu roku 2021, stejně jako o rok dříve – v březnu 2020, prožívali občané České republiky tzv. lockdown, jehož zavedení se v obou případech opíralo o analýzu dat. Těžko si představit přesvědčivější doklad prostého tvrzení, že *analýza dat významně ovlivňuje náš každodenní život.* Jak moc je analýza dat součástí našeho každodenního života snad běžně ani nevnímáme. A přece, spamové filtry v našem emailovém prohlížeči, záhadní permoníci – tzv. recommender systems (`https://en.wikipedia.org/wiki/Recommender_system`) – kteří odkudsi vykutají a zobrazí nám nabídku produktů (třeba knih), která jako by nám byla na míru ušitá, či jen „obyčejná" předpověď počasí, to jsou jen namátkově vybrané příklady výsledků analýzy dat, se kterými se setkáváme takřka denně. Čtenáři, který by si chtěl o pestrosti použití analýzy dat v běžném životě udělat lepší představu lze doporučit populárně-naučné publikace Jeffreyho S. Rosenthala [5] či Natea Silvera [6].

Prostřednictvím analýzy dat hledáme odpovědi na otázky, které nás zajímají. Jaké jsou to otázky ve výše uvedených příkladech využití analýzy dat? Poskytovatele emailových služeb zajímá, jestli příchozí email je obtěžující spam, který by měl být smazán, nebo důležitá zpráva, která naopak smazána být nesmí. Obdchodníka s knihami zajímá, kterou knihu má zákazníkovi nabídnout, aby jej nabídka zaujala. A konečně velké množství lidí zajímá, jaké bude v nejbližší době počasí, aby podle této předpovědi vhodně upravili svůj oděv (aby se tzv. přioblékli, bude-li chladněji) či dokonce program.

Ačkoliv analýza dat hraje v dnešní době důležitou roli, je třeba si neustále připomínat, že proces získávání znalostí na základě analýzy dat má svá úskalí. Je třeba být neustále ve střehu – omylům a chybám se často bohužel nevyhnou ani profesionální datoví analytici, natož poučení laici, kteří dnes mají k dispozici celou řadu softwarových nástrojů pro nejrůznější analýzy.

2 Analýzy, které „zavřely" naši zemi

Připomeňme zde prognózu, kterou tehdejší ministr vnitra ČR Jan Hamáček (`https://cs.wikipedia.org/wiki/Jan_Ham\%C3\%A1\%C4\%8Dek`) zdůvodňoval na konci února 2021 nutnost tzv. tvrdého lockdownu. Tato prognóza poskytnutá dne 24. února 2021 jako podklad pro jednání Národní ekonomické radě vlády (`https://www.vlada.cz/cz/ppov/nerv_2020/narodni-ekonomicka-rada-vlady-182438/`) byla založe-

 DOI: https://doi.org/10.11118/978-80-7509-898-6-0020

na na poměrně sofistikovaném modelu [7]. Podle modelu měl počet nových případů „kulminovat“ přibližně v půlce března, kdy měl dosahovat hodnot cca 20 tisíc nových případů denně (a to za předpokladu zavedení lockdownu na začátku března, jak k němu opravdu došlo). Realita však zdaleka nebyla tak dramatická. Jak se z dostupných dat přesvědčíme (viz níže), počty nových případů dosáhly maxima již 5. března, tedy dříve, než se mohl efekt lockdownu zavedeného 1. března projevit, a to na úrovni přibližně 12 tisíc nových případů denně. Všechna tvrzení o potřebě zavedení přísných opatření tak byla empiricky vyvrácena – k obratu ve vývoji epidemie došlo ještě předtím, než se efekt těchto opatření mohl projevit (viz text Angeliky Bazalové v časopise Reflex [1]). Poznamenejme, že tím ovšem ani není dokázáno, že by opatření neměla žádný efekt.

Na to, že predikce počtu nových případů „nevyšla“, upozornil již 11. 3. reportér Petr Holub [2]. Hlavního autor prognózy – Martin Šmíd, ve své veřejné reakci na Holubův článek [8] správně uvedl, že „Vzhledem k tomu, že jde o model stochastický (zahrnující náhodu), nelze předpokládat, že poskytne přesnou předpověď. Čáry v grafech znázorňují jen jakousi střední předpověď. Tyto bodové předpovědi by správně měly být doplněny mezemi nejistoty (konfidenčními pásy). Tyto hodnoty náš model počítá, v prezentaci pro NERV bohužel nejsou uvedeny [...]“. Selhání predikce počtu nově potvrzených případů však musel uznat: „O selhání stochastického modelu se dá mluvit až v případě, kdy se předpovědi systematicky ocitají mimo tyto pásy, což tady bohužel nastalo u počtu nových případů.“

O podstatně jednodušším modelu, který vedl k lockdownu v březnu 2020, se můžete více dozvědět v knize Pandemie [4].

2.1 Data o COVID – cvičení v R

Chceme-li se přesvědčit, jaký byl skutečný průběh počtu nově diagnostikovaných případů nákazy virem SARS-CoV-2, původcem onemocnění COVID-19, můžeme tak učinit vizualizací dat z některé z veřejně dostupných databází. V České republice počty nově diagnostikovaných případů denně zveřejňuje Ústav zdravotnických informací a statistiky ČR na stránce `https://onemocneni-aktualne.mzcr.cz/covid-19`. Stejné údaje najdeme i v mezinárodní databázi Our World in Data (`https://ourworldindata.org/coronavirus`). Datový soubor `owid-covid-data.csv` můžeme stáhnout např. z repozitáře GitHub: . (`https://github.com/owid/covid-19-data/tree/master/public/data`)

Data načteme pomocí příkazu `read.csv()` a poté vybereme jen data o České republice za období od 24. února do konce dubna 2021.

Pro práci s daty v R-ku lze v současné době doporučit balíček `dplyr`, viz [9], případně balíček `tidyr`, viz [10]. Čtenářům, kteří s těmito balíčky ještě nepracovali, doporučujeme využít některého z „taháků“, tzv. cheat sheets, které na prostoru dvou stran (tedy jednoho listu) formátu A4 přehledně shrnují to nejpodstatnější, co je k dané problematice třeba vědět. Osvědčeným tahákem je Data Wrangling with dplyr and tidyr Cheat Sheet (`https://rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf`). Novější verze, které jsou zvlášť pro balíčky `dplyr` a `tidyr`, lze nalézt v přehledu taháků . (`https://www.rstudio.com/resources/cheatsheets/`)

```
library(dplyr)

 data    = read.csv("owid-covid-data.csv")
 data.cz = data %>% filter(location=="Czechia")
 data.cz.3_4.2021 = data.cz %>% filter((date>="2021-02-24")&
                                       (date<="2021-04-30"))
```

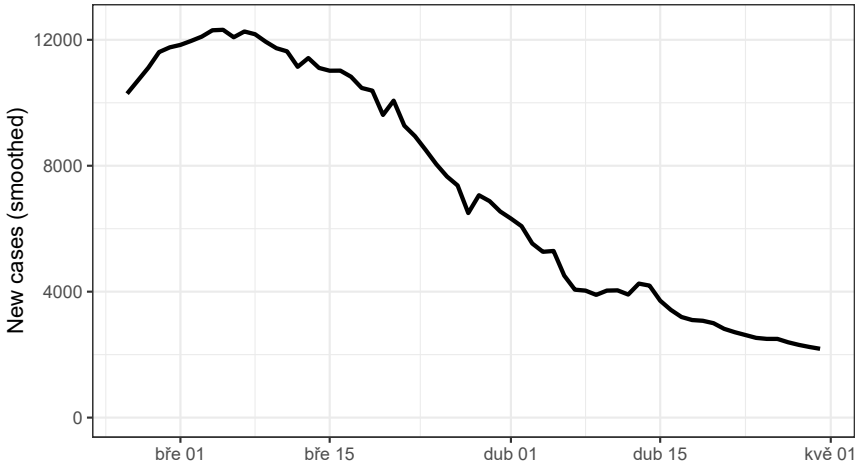
Vývoj počtu nově diagnostikovaných případů můžeme vizualizovat pomocí balíčku `ggplot2`, viz [11]. Opět doporučujeme tahák, tentokrát Data visualization with `ggplot2` cheatsheet. Budeme pracovat se sloupcem `new_cases_smoothed`. Jde o průměrné hodnoty počtu nových případů vždy za posledních 7 dní. Tímto průměrováním jsou z dat odstraněny pravidelné výkyvy s periodou jednoho týdne související s tím, že o víkendu se méně testuje a je tedy i méně zjištěných nových případů. Graf můžeme vykreslit následovně:

```
library(ggplot2)

 ggplot(data.cz.3_4.2021, aes(x=as.Date(date))) +
   geom_line( aes(y=new_cases_smoothed))
```

Obrázek 2.1 byl získán úpravou (rozšířením) tohoto základního příkazu:

```
 ggplot(data.cz.3_4.2021, aes(x=as.Date(date))) +
  geom_line( aes(y=new_cases_smoothed),size=1) +
  ylim(c(0,12500)) +
  labs(y="New cases (smoothed)", x="") +
  theme_bw()
```



Obr. 1: *Průměrné denní počty nově diagnostikovaných případů nákazou virem SARS-CoV-2 v České republice v období 24. 2. 2021 až 30. 4. 2021 (7-denní průměry).*

3 Je lépe chválit nebo kárat?

Je dobré si uvědomit, že proces získávání znalostí z dat není jen něčím, co dělají datoví analytici. To že si utváříme názor na základě svých pozorování (tedy „dat" ukládaných do naší paměti), je přece přirozené. Také v tomto procesu se však naše mysl dopouští mnoha chyb, jak popisuje v knize *Myšlení: rychlé a pomalé* [3] psycholog Daniel Kahneman.

V části nazvané Heuristiky a zkreslení věnoval Kahneman jednu kapitolu jevu nazývanému regrese k průměru (viz [3], str. 189–199). Kapitolu uvádí příhodou z doby, kdy v rámci kuzu psychologie vysvětloval instruktorům izraelského vojenského letectva zásadu, že „odměna za zlepšený výkon funguje lépe než trest za chybný výkon". Jeden ze zkušených instruktorů však byl opačného názoru. Argumentoval takto: „Vždycky jsem si dával záležet, abych lidi za pěkně provedený manévr pochválil, a příště ho vždycky provedli hůř. Ale když jsem je za špatně provedený manévr seřval, většinou se pak zlepšili. Tak mi netvrďte, že chvála pomáhá a trestání ne. Moje zkušenost je přesně opačná." Kahneman uvádí, že byť byla instruktorova pozorování správná, závěr ohledně účinnosti odměny a trestu z těchto pozorování vyvozený byl naprosto špatný. Nezpochybnitelné vysvětlení instruktorovy zkušenosti je známo jako *regrese k průměru*. Tento princip nyní objasníme.

3.1 Fenomén regrese k průměru a jeho ilustrace

Představme si, že dostaneme za úkol se v krátkém časovém intervalu naučit 100 pro nás zatím neznámých slovíček v cizím jazyce. Zvládneme jen 80, zbylých 20 bohužel ne. Jak asi dopadneme v testu, kde máme přeložit 20 slovíček náhodně vybraných ze zadané stovky slov? Umíme 80 ze 100, tedy 80 % všech slovíček, takže bychom očekávali 80% úspěšnost i v testu. Té dosáhneme, když z 20 slovíček správně přeložíme 16. Může se ale stát, že budeme mít štěstí a z vybraných slovíček správně přeložíme více než 16. Jelikož umíme přeložit 80 slovíček, může se dokonce stát, že správně přeložíme všech 20 slovíček. Při hodně velké smůle by se nám však teoreticky mohlo stát, že budeme tázáni právě na těch zbylých 20 slovíček, která neumíme. To by však musela být opravdu neuvěřitelně velká smůla! Jak velká? To umíme spočítat. předpokladu, že všechna slovíčka mají stejnou šanci, že budou vybrána, se

Počet vybraných slovíček, které umíme přeložit, se řídí tzv. *hypergeometrickým rozdělením*. Pravděpodobnost, že jich bude právě x, kde za x můžeme dosadit libovolné celé číslo od nuly po 20, je rovna

$$p(x) = \frac{\binom{80}{x} \cdot \binom{20}{20-x}}{\binom{100}{20}}.$$

Dosadit do tohoto vzorce postupně všechny možné hodnoty x $(0, 1, 2, \ldots, 20)$ můžeme v R-ku jediným příkazem, a to s využitím funkce `dhyper()`:

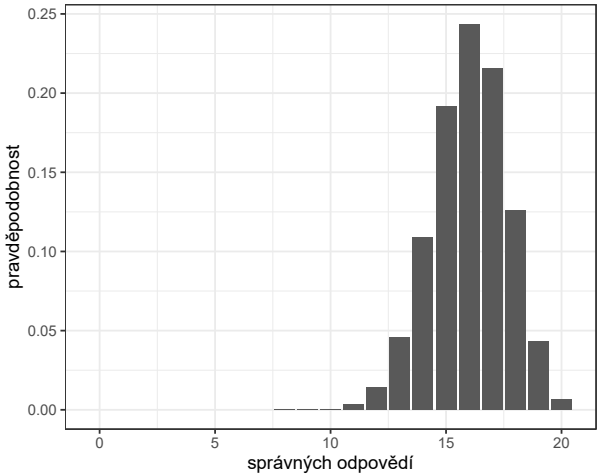
```
dhyper(0:20,80,20,20)
```

Vypočtené pravděpodobnosti můžeme pro přehlednost zobrazit prostřednictvím sloupcového grafu, viz obrázek 3.1. Uvádíme zde kód potřebný k jeho vygenerování v softwaru R (opět používáme balíček `ggplot2`).

```
pocet.spravnych = 0:20
pravdepodobnost = dhyper(0:20,80,20,20)
data = data.frame(pocet.spravnych,pravdepodobnost)

ggplot(data,aes(pocet.spravnych,pravdepodobnost)) +
   geom_col()+
   labs(x = "správných odpovědí", y = "pravděpodobnost") +
   theme_bw()+
```

Na obrázku 3.1 vidíme, že vskutku nejpravděpodobnějším výsledkem testu je zisk 16 bodů z 20. Možná nás však překvapí, že pravděpodobnost tohoto výsledku je „jen“ asi



Obr. 2: *Rozdělení pravděpodobnosti počtu správně přeložených slovíček.*

24 %. Dosti pravděpodobný je také zisk 17 bodů (asi 21 %) nebo 15 bodů (asi 19 %). Více než 10% pravděpodobnost má taky zisk 18, resp. 14 bodů. Výše zmiňovaná možnost, že bychom uměli správně přeložit všech 20 slovíček, má celkem malou pravděpodobnost (0,66 %). Rovněž vidíme, že správné přeložení pouze 10 či dokonce ještě méně vybraných slovíček je velmi nepravděpodobné a muselo by tedy být pokládáno za opravdu velkou nepřízeň štěstěny.

Představme si nyní školní třídu, která má 30 žáků. Každý z těchto žáků se naučil právě 80 ze 100 slovíček[1].

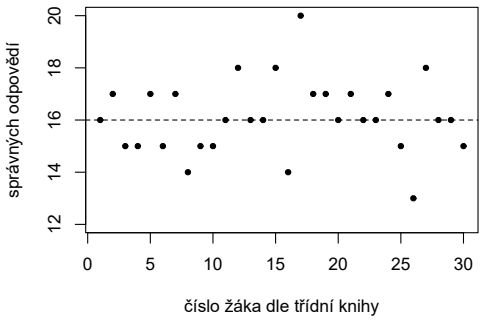
Jak asi dopadnou v testu připravenosti, v němž budou mít za úkol přeložit 20 náhodně vybraných slovíček? Výše vypočtené pravděpodobnosti naznačují, že přibližně čtvrtina žáků (7 až 8) by měla mít 16 správně přeložených slov. Hodně bude i žáků s 15 či 17 správnými překlady, neboť součet $p(15) + p(16) + p(17) = 0{,}19 + 0{,}24 + 0{,}21 = 0{,}64$; čekáme tedy, že kolem dvou třetin žáků (tj. dvacet) by mělo mít výsledek v rozmezí 15 až 17 správných odpovědí.

[1] Snad si můžeme dovolit předpokládat, že slůvka jsou pro zapamatování přibližně stejně obtížná. Debata o tomto předpokladu by jistě byla zajímavá a mohla by přinést vylepšení uvažovaného jednoduchého modelu. K tomu bychom však potřebovali nasbírat skutečná data.

Pomocí generátoru (pseudo)náhodných čísel můžeme pro každého žáka vygenerovat počet získaných bodů. Generujeme z hypergeometrického rozdělení, proto použijeme příkaz `rhyper()`. Data pak vizualizujeme.

```
set.seed(8)
test1 = rhyper(30,80,20,20)

plot(test1,pch=20,
   xlim=c(1,30), ylim=c(16-4,16+4),
     ylab="správných odpovědí", xlab="číslo žáka")
abline(h=16,lty=2)
```



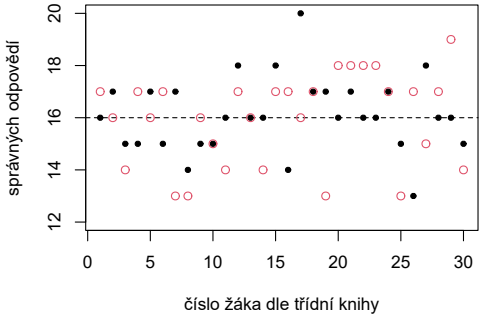
Obr. 3: *Výsledky 30 stejně dobře připravených žáků v testu o 20 otázkách.*

Na obrázku 3.1 můžeme porovnat svá výše popsaná očekávání s „realitou“. Potvrdilo se, že nejčastějším výsledkem je 16 správných odpovědí. Tohoto výsledku dosáhlo 9 žáků. Nejlepší výsledek dosáhl žák, který je v třídní knize veden pod číslem 17. Měl správně všech 20 odpovědí. Naopak nejhůře dopadl žák č. 26, který měl jen 13 správných odpovědí. Uvědomme si, že přitom oba tito žáci (stejně jako i všichni ostatní žáci) byli připraveni stejně dobře (uměli 80 ze 100 slovíček). Jen jeden měl více štěstí (a získal o 4 body více, než by odpovídalo jeho schopnostem) a druhý měl velkou porci smůly (a získal o 3 body méně, než odpovídá jeho schopnostem). Vyučující by si měl být vědom toho, že počet správných odpovědí, kterého žák v testu dosáhl, často není přesným obrazem žákových schopností (znalostí), ale je do značné míry určován náhodou. Kdo z nás by předpokládal, že žák, který dosáhl 20 bodů z 20 možných, byl ve skutečnosti stejně dobře připraven jako ten, který měl jen 13 bodů z 20? Jak správně porozumět dosaženým výsledkům, říká následující „rovnice“:

$$\begin{aligned} \text{výsledek} &= \text{schopnost} + \text{štěstí} \\ 20 \text{ bodů} &= 16 \text{ bodů} + 4 \text{ body} \\ 13 \text{ bodů} &= 16 \text{ bodů} - 3 \text{ body} \end{aligned}$$

Představme si nyní, že se vyučující rozhodne s žáky, z nichž každý stále umí přesně 80 ze 100 slovíček, napsat ještě jeden test o 20 položkách. Opět vybírá náhodně 20 ze 100 slov, přitom nezáleží na tom, jestli už slovo bylo vybráno v prvním testu. Vygenerujme opět výsledky žáků a zobrazme je do grafu spolu s výsledky prvního testu. Můžeme to udělat tak, že do původního grafu přidáme červeně znázorněné symboly odpovídající výsledkům druhého testu.

```
test2 = rhyper(30,80,20,20)
points(test2,col=2,pch=21,cex=1)
```



Obr. 4: *Výsledky 30 stejně dobře připravených žáků v prvním (černě) a druhém (červeně) testu o 20 otázkách.*

Srovnání výsledku prvního a druhého testu vidíme na obrázku 3.1. Z něj je patrné, že ti, kterým v prvním testu štěstí přálo nejvíc (dosáhli extrémně dobrého výsledku), již podruhé tolik štěstí neměli – nejlepší žák si pohoršil z 20 bodů na 16, ani nikdo ze tří žáků, kteří měli 18 bodů, tento výkon při druhém testu nezopakoval. Naproti tomu žák s nejhorším výsledkem v prvním testu si ve druhém polepšil ze 13 bodů na 17. Je dobré si povšimnout i toho, že žák s poměrně dosti špatným výsledkem prvního testu (žák č. 8)

měl ve druhém testu ještě větší smůlu a pohoršil si ze 14 bodů na 13. Tento případ je však spíše výjimečný. Ukazuje, že „návrat k průměru“, který jsme pozorovali v ostatních případech, kdy po extrémně dobrém výsledku došlo k zhoršení a po extrémně špatném došlo ke zlepšení (aniž by se měnily schopnosti žáků), je pravděpodobný, nikoliv však nevyhnutelný.

Všimněme si, že ke skutečnému zlepšení či zhoršení schopností studentů nedošlo. Stále jsme uvažovali situaci, kdy všichni umí přesně 80 ze 100 slovíček. Pozorované zhoršení (výsledek druhého testu byl horší než výsledek prvního) či naopak zlepšení, bylo tedy jen zdánlivé a bylo „způsobeno“ vlivem náhody (štěstí) na výsledek testu. Učitel, který by žáky s nejvyšším počtem bodů v prvním testu pochválil a naopak ty nejhorší by pokáral, a pak pozoroval „negativní efekt“ pochvaly a „pozitivní efekt“ pokárání, se bude v hodnocení ekeftivity svého zásahu (pochvaly, resp. pokárání) mýlit, jako se mýlil instruktor letectva vzpomenutý na začátku této kapitoly.

Čtenáře, který stále ještě není přesvědčen o zdánlivosti efektu pochvaly a pokárání, vybízíme: Vezměte hrací kostku a tu pochvalte vždy, když padne šestka. Uvidíte, že v příštím hodu se nezlepší, častěji se dokonce zhorší. A taky té kostce vynadejte, když padne jednička. Uvidíte, jak se v dalším hodu zlepší (nebo alespoň nezhorší)! Opravdu věříte, že to zhoršení (resp. zlepšení) je kvůli (díky) tomu, že kostku chválíte (káráte)?

Kahneman výše popsaný problém v úsudku ohledně efektivity pochval a kárání komentoval takto: „Zpětná vazba, které nás život vystavuje, je zvrácená. Protože máme tendenci být na jiné lidi milí, když nás potěší, a nepříjemní, když nás nepotěší, ze statistického hlediska jsme trestáni za to, že jsme milí, a odměňováni za to, že jsme nepříjemní.“

4 Závěr

V tomto příspěvku jsme se snažili poukázat na některá úskalí procesu získávání znalostí na základě analýzy dat. Připomněli jsme selhání predikce (podložené daty) sofistikovaného modelu pro předpověď počtu nově diagnostikovaných infekcí virem SARS-CoV-2 z března 2021. Na příkladu hodnocení efektu pochvaly a kárání jsme pak ukázali, jak obtížným úkolem je hodnocení efektu našeho konání a k jakým omylům je naše mysl náchylná. Možná nám to pomůže pochopit obtíže, které máme při vyhodnocování efektivity různých restrikcí, jejichž cílem je různě ovlivňovat rychlost šíření nákazy. Pravdou je, že restrikce se zavádí v situaci, kdy počty nakažených (drameticky) rostou, a naopak „rozvolňuje se“, když epidemie ustupuje. Znamená to však, že opatření fungují? Ne nutně. Tím ovšem netvrdíme, že nefungují. Jen upozorňujeme, že to, co se někomu zdá být zřejmé, zdaleka zřejmé být nemusí.

Snažili jsme se náš výklad doplnit o ukázky práce se softwarem R. Pokud se čtenář dozvěděl něco nového o jeho použití, tím lépe.

Literatura

1. Bazalová, A. Jenom lockdowny virus nezastaví. *Reflex*, č. 12, roč. 2021, str. 9–11.
2. Holub, P. *Hamáčkova předpověď o 20 tisících nakažených nevyšla. Proč?* [online]. 11.3.2021 [cit. 2.11.2021]. Dostupné z:8 . (`https://www.seznamzpravy.cz/clanek/hamackova-predpoved-o-20-tisicich-nakazenych-nevysla-proc-146362`)
3. Kahneman, D. *Myšlení: rychlé a pomalé*. Brno: Jan Melvil, 2012. Pod povrchem. ISBN 978-80-87270-42-4.
4. Kubal, M., Gibiš, V. *Pandemie*. Praha: Kniha Zlín, 2020. ISBN 978-80-7662-047-6.
5. Rosenthal, J. S. *Zasažen bleskem: podivuhodný svět pravděpodobností*. Praha: Academia, 2008. Galileo. ISBN 978-80-200-1645-4.
6. Silver, N. *Signál a šum: mnoho předpovědí selže, některé ne*. Praha: Paseka, 2014. ISBN 978-80-7432-440-6.
7. Šmíd, M. et al. *SEIR Filter: A Stochastic Model of Epidemics*. medRxiv, 2021. Dostupné z doi: 10.1101⊠.02.16.21251834
8. Šmíd, M. *Ještě o Hamáčkově předpovědi...* [online]. 12.3.2021 [cit. 2.11.2021]. Dostupné z: `https://www.bisop.eu/jeste-o-hamackove-predpovedi/`
9. Wickham, H., François, R., Henry, L., Müller, K. (2021). dplyr: A Grammar of Data Manipulation. R package version 1.0.7. `https://CRAN.R-project.org/package=dplyr`
10. Wickham, H. (2021). tidyr: Tidy Messy Data. R package version 1.1.4. `https://CRAN.R-project.org/package=tidyr`
11. Wickham, H. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4.

Autor

Mgr. Ondřej Vencálek, Ph.D., Česká statistická společnost, Na padesátém 81, 100 82 Praha 10

Fourierovy řady s podporou systému počítačové algebry wxMaxima

Aleš Kozubík

Abstrakt: V příspěvku je prezentována pokročilejší ukázka použití systému počítačové algebry `wxmaxima` při podpoře výuky matematické analýzu. Nejprve se stručně charakterizuje rozvoj funkce do Fourierovy řady, řady sinů a řady kosinů. tento postup je následně implementován v podobě skriptu prostředí `wxmaxima`. V závěru se koncentrujeme na pokročilejší ukázku s nalezením funkčního předpisu zadaného vzorku a jeho rozvinutí do periodického pokračování.

Abstract: This paper presents a more advanced demonstration of the use of the computer algebra system `wxmaxima` in supporting the teaching of mathematical analysis. First, the development of a function into the Fourier series, the series of sines, and the series of cosines is briefly characterized. Consequently, this procedure is implemented as a script in the `wxmaxima` environment. Finally, we concentrate on a more advanced demonstration of finding the functional prescription of a given sample and developing it into a periodic continuation.

1 Úvod

Fourierovy řady se používají se zejména při studiu jevů s periodickým charakterem. Tato metoda byla původně vyvinuta pro hledání periodických řešení diferenciálních rovnic. Dnes však nacházejí daleko širší uplatnění. rozklad na vlny se přirozeně objevuje například při ukládání zvuku do audio souborů, při kompresi hudby, kde tvoří základ formátu `mp3`. rovněž lze zmínit kompresi a filtrování obrazů a v neposlední řadě počítačové zpracování signálů.

Proto nikoho nepřekvapí, že problematika rozvoje funkcí do Fourierovy řady se stala organickou součástí kurzu matematické analýzy pro technické obory, informatiku nevyjímaje. Jak uvidíme v dalším textu, svojí povahou jsou vhodné pro řešení s využitím počítačů. koeficienty rozvoje jsou dány jednoduchými vzorci, jež lze snadno programovat v systémech počítačové algebry s podporou symbolických výpočtů.

V první části příspěvku uvedeme stručný úvod do problematiky Fourirerových řad se zaměřením na Fourierovy řady vzhledem ke trigonometrickému systému funkcí. Vzhledem k tomu, že pro podporu výuky matematické analýzy jsme vybrali systém počítačové algebry `wxmaxima`, který je distribuován pod svobodnou licencí s otevřeným kódem, budeme si ilustrovat jednoduchý způsob naprogramování rozvoje v tomto systému. O něco pokročilejší přístup si pak v závěrečné části přiblížíme na úloze reprezentovat

 DOI: https://doi.org/10.11118/978-80-7509-898-6-0030

periodický rozvoj pro zadaný daný signál, aniž by byl znám jeho funkční předpis. Jde tedy o co nejbližší přiblížení úlohy reálné situaci v praxi.

Všechny uvedené zdrojové kódy lze aplikovat bez hlubší znalosti prostředí systému `wxmaxima`. Lze však doporučit bližší seznámení s tímto systémem například prostřednictvím publikací [3] a [4] nebo [5].

2 Fourierova řada

Pojem Fourierovy řady lze ve všeobecnosti zavést v libovolném abstraktním prostoru, ve kterém je definován skalární součin a je známa jeho ortogonální resp. ortonormální báze. My se od těchto abstraktních konstrukcí odchýlíme a budeme se věnovat specifickému, ale nejznámějšímu případu rozvoje vzhledem k ortonormálnímu trigonometrickému systému funkcí. Tento je tvořen posloupností funkcí ve tvaru

$$\frac{1}{\sqrt{2\pi}}, \frac{1}{\sqrt{\pi}}\cos x, \frac{1}{\sqrt{\pi}}\sin x, \frac{1}{\sqrt{\pi}}\cos 2x, \frac{1}{\sqrt{\pi}}\sin 2x, \ldots\ldots, \frac{1}{\sqrt{\pi}}\cos nx, \frac{1}{\sqrt{\pi}}\sin nx, \ldots \tag{1}$$

Je-li dána funkce $f(x)$, jež je integrovatelná na intervalu $\langle -\pi, \pi\rangle$, pak její rozvoj do Fourierovy řady vzhledem k ortonormálnímu systému 1 má tvar

$$\frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n \cos nx + b_n \sin nx), \tag{2}$$

kde a_n,b_n jsou Fourierovy koeficienty, pro něž platí

$$\begin{aligned} a_n &= \tfrac{1}{\pi}\textstyle\int_{-\pi}^{\pi} f(x)\cos nx\,\mathrm{d}x, \quad n \in \mathbb{N}\cup\{0\} \\ b_n &= \tfrac{1}{\pi}\textstyle\int_{-\pi}^{\pi} f(x)\sin nx\,\mathrm{d}x, \quad n \in \mathbb{N} \end{aligned}$$

Na tomto místě poznamenejme, že uvedené vztahy platí nejen na intervalu $\langle -\pi, \pi\rangle$, ale na libovolném intervalu $\langle c, c+2\pi\rangle$délky 2π. Samozřejmě, použití si v takovém případě vyžaduje příslušnou změnu integračních mezí dle hranic intervalu.

Fourierovu řadu definovanou rovnicí 2 lze snadno transformovat na periodické funkce s libovolnou periodou $p = 2h$. Je-li totiž $f(x)$ periodická funkce s periodou $p = 2h$, která je integrovatelná na intervalu $\langle -h, h\rangle$, pak funkce $g(t) = f\left(\frac{h}{\pi}x\right)$ je periodická funkce s periodou 2π integrovatelná na intervalu $\langle -\pi, \pi\rangle$. Tuto funkci lze dle vztahu 2 rozvinout do Fourierovy řady na intervalu $\langle -\pi, \pi\rangle$. Zpětnou transformací $x = \frac{\pi}{h}t$ získáme Fourierovu řadu funkce $f(x)$ na intervalu $\langle -h, h\rangle$. Fourierova řada funkce $f(x)$ na intervalu $\langle -h, h\rangle$ má tudíž tvar:

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos \frac{n\pi}{h} x + b_n \sin \frac{n\pi}{h} x), \tag{3}$$

kde Fourierovy koeficienty a_n a b_n jsou dány vztahy

$$\begin{aligned} a_n &= \frac{1}{h} \int_{-h}^{h} f(x) \cos \frac{n\pi}{h} x \, \mathrm{d}x, \quad n \in \mathbb{N} \cup \{0\} \\ b_n &= \frac{1}{h} \int_{-h}^{h} f(x) \sin \frac{n\pi}{h} x \, \mathrm{d}x, \quad n \in \mathbb{N} \end{aligned}$$

I zde upozorněme na skutečnost, že uvedený rozvoj platí na libovolném intervalu $\langle c, c+2h\rangle$ délky $2h$. Rovněž zde je nutno transformovat integrační meze dle hranic uvedeného intervalu.

Rozšířením Fourierových řad je rozvoj do řady sinů resp. do řady kosinů. Pro tyto účely je nutno definovat pojem sudého a lichého rozšíření funkce. Nechť $f(x)$ je funkce integrovatelná na $\langle 0, h\rangle$ [resp. $(0, h\rangle$]. Položíme-li pro $x \in \langle -h, 0\rangle$ $f(x) = f(-x)$ [resp. $f(x) = -f(-x)$ a $f(0) = 0$], tak pravíme, že jsme sestrojili sudé (resp. liché) rozšíření funkce $f(x)$ na interval $\langle -h, h\rangle$.

Máme-li definováno sudé a liché rozšíření funkce, lze snadno charakterizovat rozvoj do řady sinů resp. kosinů. Fourierova řada sudého (lichého) rozšíření funkce $f(x)$ se nazývá řadou kosinů (řadou sinů) funkce $f(x)$ na intervalu $\langle 0, h\rangle$. V takovém případě lze vztahy pro Fourierovy koeficienty 3 zjednodušit do následující podoby.

Nechť $f(x)$ je funkcie integrovatelná na intervalu $\langle -h, h\rangle$. Je-li je tato funkce sudá, pak její Fourierova řada má tvar

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos \frac{n\pi}{h} x, \tag{4}$$

kde $a_n = \frac{2}{h} \int_0^h f(x) \cos \frac{n\pi}{h} x \, \mathrm{d}x$. Je-li tato funkce lichá, pak její Fourierova řada má tvar

$$\sum_{n=1}^{\infty} b_n \sin \frac{n\pi}{h} x, \tag{5}$$

kde $b_n = \frac{2}{h} \int_0^h f(x) \sin \frac{n\pi}{h} x \, \mathrm{d}x$.

Podrobnější výklad problematiky Fourierových řad včetně podmínek jejich konvergence lze nalézt například v [2], [6] nebo ve slovenském jazyce v [1].

3 Ukázka řešení ve wxmaxima

Je-li dán předpis funkce, již chceme rozvinout, je poměrně jednoduché v prostředí systému wxmaxima naprogramovat rozvoj do Fourierovy řady. Programový „opis“ vztahů 2 resp. 3 zvládne prakticky každý, kdo je alespoň zběžně obeznámen se syntaxí tohoto prostředí. Zde si postup ilustrujeme na případu rozvoje funkce $f(x)^2$ na intervalu $\langle -2, 2\rangle$. Tvorbu příslušného skriptu pojmeme poněkud šířeji, aby ho bylo možno opakovaně použít i pro jiné funkce na libovolném intervalu. Při programování se budeme pro lepší orientaci přidržovat označení ze vztahů 2 resp. 3.

```
(declare(n,integer),assume(n>0),facts());
```

Dále je nutno definovat délku půlperiody h, začátek intervalu, ba němž určujeme rozvoj a uložíme ho v proměnné start a samotnou funkci f(x), kterou hodláme do Fourierovy řady rozvíjet.

```
h:2$
start:-2$
f(x):=x^2$
```

Jsou-li nyní určeny všechny hodnoty nezbytné pro výpočet, lze definovat funkce k určení koeficientů Fourierova rozvoje dle vztahu 3. Vzhledem k tomu, že pole jsou v systému wxmaxima indexována od počáteční hodnoty 1, musíme koeficient a_0 definovat samostatně, jak vidíme z následujícího kódu:

```
a0:integrate(f(x),x,start,start+2*h)/h;
define(a(n),integrate(f(x)*cos(n*x*%pi/h),x,start,start+2*h)/h);
define(b(n),integrate(f(x)*sin(n*x*%pi/h),x,start,start+2*h)/h);
```

S využitím zavedených funkcí pro Fourierovy koeficienty lze definovat rozvoj do Fourierovy řady následujícím kódem:

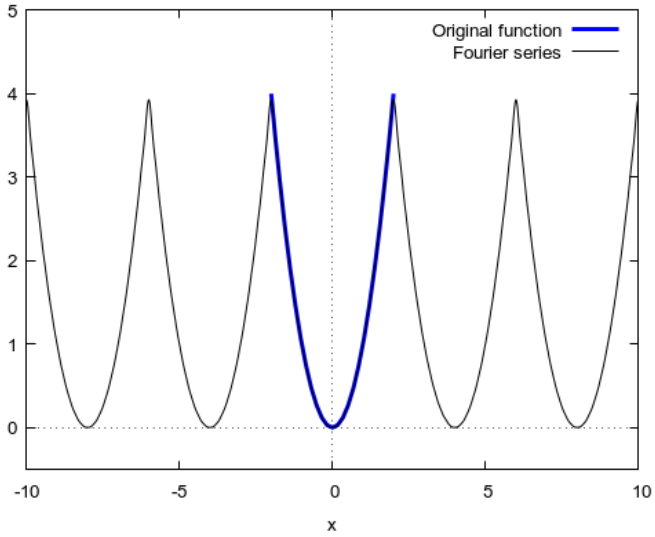
```
frada(nmax):=a0/2+sum(a(m)*cos(m*%pi*x/h)+b(m)*sin(m*%pi*x/h),m,1,nmax);
```

Jedinou proměnnou této funkce je nmax, která udává délku rozvoje. Jejím zadáním s příslušnou hodnotou obdržíme analytické vyjádření požadovaného rozvoje. Výsledek si lze pomocí funkce plot2d() resp. wxplot2d() prezentovat graficky. Příslušný kód má tuto podobu:

```
g(x):=if x>start and x<start+2*h then f(x);
wxplot2d([g(x),frada(20)],[x,-10,10],[y,-0.5,5],
[legend,"Original function","Fourier series"],[style,[lines,3,1],
[lines,1,5]]),wxpolt_size=[600,400];
```

```
plot2d([g(x),frada(20)],[x,-10,10],[y,-1,5],[style,[lines,3,1],
[lines,1,5]],[legend,"Original function","Fourier series"]);
```

Poznamenejme, že definice funkce `g(x)` slouží k zobrazení původního vzorku a je nepovinná, pokud nechceme porovnávat rozvoj a původní funkci (je pak ale rovněž nutno odstranit funkci `g(x)` se seznamu funkcí zobrazovaných pomocí `plot2d()` resp. `wxplot2d()`). Výsledný graf vidíme na obrázku 1.



Obr. 1: *Zobrazení Fourierovy řady funkce* $f(x) = x^2$ *na intervalu* $\langle -2, 2 \rangle$.

4 Ukázka pokročilejší práce

V realitě se nevyskytují informační zdroje, jež by poskytovaly informaci o explicitním funkčním předpisu analyzovaného vzoru. naopak, musíme sami usilovat o jejich nalezení. Tuto situaci během laboratorních cvičení simulujeme zadáním grafického vzorku, který mají studenti za úlohu nejdříve vyjádřit jako funkci resp. jako křivku poskládanou z fragmentů několika funkcí, a následně ji rozvinout do Fourierovy řady, řady sinů a řady kosinů.

Zkušenosti z vyučování nám ukázaly, že největší potíže činí právě nalezení tohoto funkčního předpisu, byť pro jednoduchost předpokládáme, že vzorek je tvořen jen lineárními segmenty, tedy úseků poskládaných z přímek. Připomeňme tedy, jak nalézt

předpis pro danou úsečku resp. rovnici přímky, z níž je úsečka vyťata. Situace je znázorněna na obrázku 2.

Při řešení tohoto problému vycházíme z nutnosti nalézt hodnotu koeficientů a a b z rovnice přímky $y = ax + b$. Souřadnice krajních bodů úsečky musí vyhovovat této rovnici a jejich dosazením tak získáme soustavu dvou rovnic pro neznámé koeficienty a a b. Tato soustava má (při dodržení označení dle obrázku 2) tvar:

$$y_1 = ax_1 + b$$
$$y_2 = ax_2 + b$$

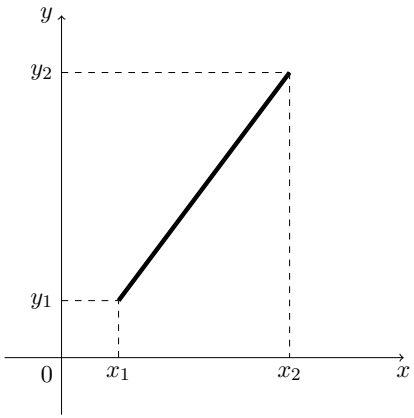
Jejím řešením obdržíme vztahy pro koeficienty a a b, jichž lze použít pro libovolný segment zadaného vzorku. Toto vyjádření má tvar:

$$a = \frac{y_2 - y_1}{x_2 - x_1}$$
$$b = y_1 - \frac{y_2 - y_1}{x_2 - x_1} x_1$$

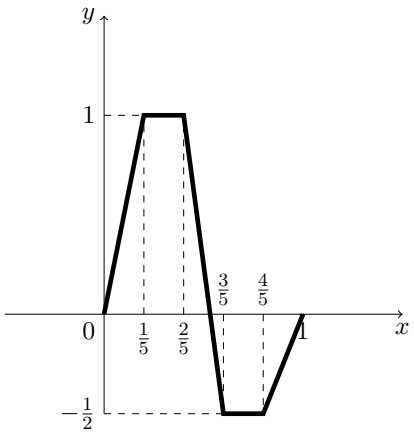
Ukázka možného zadání pro hledání Fourierovy řady je na obrázku 3. V tomto konkrétním případě sestává z pěti lineárních segmentů, jejichž koncové body zadáme do systému jako seznam o délce 6 prvků. Protože chceme úlohu řešit co nejobecněji, zjišťujeme délku seznamu v rámci kódu a pak cyklicky určujeme rovnice jednotlivých fragmentů. Výsledná vyjádření ukládáme jako seznam v proměnné `useky`. Zodpovídající kód má pak následující tvar:

```
kill(all);
seznam:[[0,0],[1/5,1],[2/5,1],[3/5,-1/2],[4/5,-1/2],[1,0]]$
delka:length(seznam)$
h:(seznam[delka][1]-seznam[1][1])/2$
for j:1 while j<delka do
(
a[j]:=(seznam[j+1][2]-seznam[j][2])/(seznam[j+1][1]-seznam[j][1]),
b[j]:=(seznam[j][2]*seznam[j+1][1]-seznam[j+1][2]*seznam[j][1])/
(seznam[j+1][1]-seznam[j][1]))$
useky:makelist(a[i]*x+b[i],i,1,delka-1);
```

Nyní již lze přistoupit k samotnému určení Fourierových koeficientů dle vztahů 3 resp. dle 5 prožadu sinů a 4 pro řadu kosinů. Důležité je uvědomit si techniku výpočtu příslušných integrálů a integrační cestu rozdělit na jednotlivé intervaly, tak jak jsou určeny dělícími body zadané křivky. Opět je nutno nejprve samostatně spočíst koeficient a_0 a pak definovat koeficienty číslované od indexu 1. Příslušný zdrojový kód pak nabude tvaru:



Obr. 2: *Lineární segment – úsečka spojující body* $[x_1, y_1]$ *a* $[x_2, y_2]$.



Obr. 3: *Zadání vzorku z lineárních segmentů pro rozvoj do Fourierovy řady.*

```
a0:(1/h)*sum(integrate(useky[j],x,seznam[j][1],seznam[j+1][1]),
j,1,delka-1);
define(a(n),(1/h)*sum(integrate(useky[j]*cos(n*x*%pi/h),x,
seznam[j][1],seznam[j+1][1]),j,1,delka-1));
define(b(n),(1/h)*sum(integrate(useky[j]*sin(n*x*%pi/h),x,
seznam[j][1],seznam[j+1][1]),j,1,delka-1));
frada(nmax):=a0/2+sum(a(m)*cos(m*%pi*x/h)+b(m)*sin(m*%pi*x/h),m,1,nmax);
```

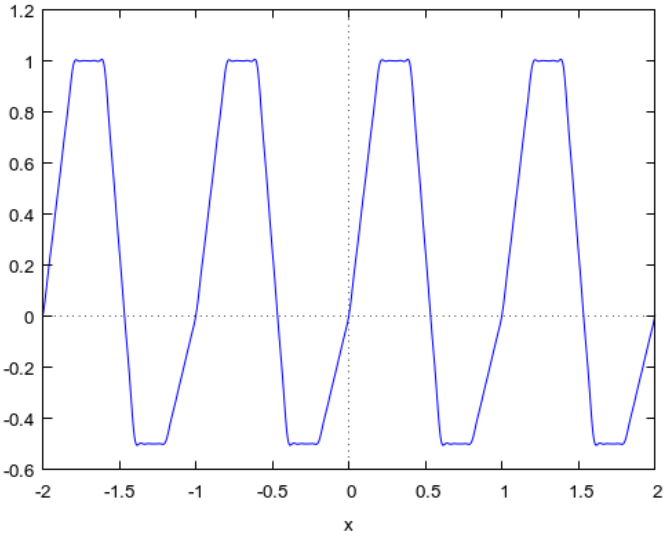
Výslednou Fourierovu řadu pak zobrazíme pomocí funkce `wxplo2dt()` resp. `plot2d()` s výsledkem prezentovaným na obrázku 4.

Dále definujeme funkce pro výpočet koeficientů pro příslušné řady sinů a kosinů. Pro odlišení od původních Fourierových koeficientů jsou označeny jako `bn` resp `ap`.

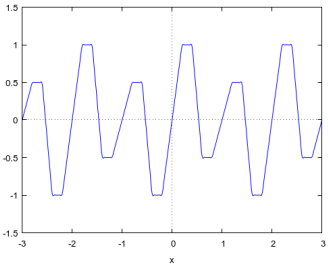
```
define(bn(n),(1/h)*sum(integrate(useky[j]*sin(n*x*%pi/(h*2)),x,
seznam[j][1],seznam[j+1][1]),j,1,delka-1));
ap0:(1/h)*sum(integrate(useky[j],x,seznam[j][1],
seznam[j+1][1]),j,1,delka-1);
define(ap(n),(1/h)*sum(integrate(useky[j]*cos(n*x*%pi/(h*2)),x,
seznam[j][1],seznam[j+1][1]),j,1,delka-1));
```

Po jejich výpočtu již efinujeme funkce pro řady sinů a kosinů ve tvaru:

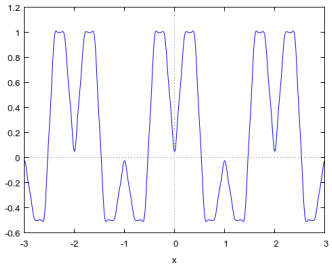
```
radasinu(nmax):=sum(bn(m)*sin(m*%pi*x/(h*2)),m,1,nmax);
```



Obr. 4: *Zobrazení Fourierovy řady vzorku na obrázku 3.*



Obr. 5: *Řada sinů pro vzorek dle obrázku 3.*



Obr. 6: *Řada sinů pro vzorek dle obrázku 3.*

```
radakosinu(nmax):=ap0/2+sum(ap(m)*cos(m*%pi*x/(h*2)),m,1,nmax);
```

Výsledky pak vidíme na obrázcích 5 a 6.

Literatura

1. Demetrian, M. *Fourierove rady a Fourierovv integrál* , Bratislava, Univezita Komenského, 2011.

2. Došlá, Z., Novák, V. *Nekonečné řady*, Brno, Masarykova Univezita, 2007.

3. Hannan, Z. *WxMaxima for Calculus I*, Solano, Comunty College, 2015.

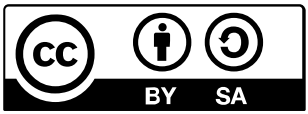
4. Hannan, Z. *WxMaxima for Calculus I*, Solano, Comunty College, 2015.

5. Kanagasabapathy, M. *Introduction to wxMaxima for Scientific Computations*, New Delhi, BPB Publications, 2018.

6. Kufner, A., Kadlec, J. *Fourierovy řady*, Brno, Academia, 1969.

Autor

RNDr. Aleš Kozubík, PhD., Katedra matematických metód a operačnej analýzy, Fakulta riadenia a informatiky, Žilinská univerzita v Žiline, Vysokoškolákov 8215/1, 010 26 Žilina, Slovenská republika, e-mail: alesko@frcatel.fri.uniza.sk



Předměty a jejich koncepce

Courses and their concepts

Pravděpodobnost a statistika s programováním v R

Aleš Kozubík

Abstrakt: Příspěvek je věnován přípravě základního kurzu pravděpodobnosti a statistiky s podporou open source programovacího prostředí R. Stručně představuje předpokládanou obsahovou náplň připravovaného kurzu s důrazem na programování v R. Autor v článku uvádí zásadní důvody pro výběr právě tohoto prostředku. V druhé části pak ilustruje vyžití nástroje na práci s aktuálními reálnými daty.

Abstract: The paper is concerned with preparing an elementary course in probability and statistics with the support of the open-source programming environment R. It briefly presents the expected content of the planned curriculum with an emphasis on programming in R. The author presents the main reasons for choosing this particular tool. In the second part, he illustrates the use of this measure to work with actual real data.

1 Úvod

Znalost problematiky matematické statistiky je nezbytná pro všechny technické, ekonomické ale i přírodovědné či humanitní odbory studia. I když se ve většině případů snažíme přírodní zákonitosti postihnout deterministickými modely a exaktními zákonitostmi, nelze se statistickým šetřením vyhnout. Existuje totiž mnoho procesů, jenž nejsme schopni, ať již pro jejich složitost nebo velké množství faktorů, které je ovlivňují, popsat deterministickými zákonitostmi. V takových případech se odvoláváme na působení náhodnosti a na naše historické empirické zkušenosti. Ale i všechny přírodní zákonitosti, které se dnes učíme jako deterministické popsané matematickými vztahy se rodily na základě pozorování a experimentování. jsou tedy rovněž výsledkem statistických metod, ač si to již při jejich prezentaci neuvědomujeme. Jako příklady procesů, jimž stále přisuzujeme nahodilý charakter, uveďme alespoň několik z nich.

Typickým příkladem jsou modely rizika ve financích a pojišťovnictví. Zejména v pojištění, ať již životním nebo neživotním, se smlouvy vážou na události náhodného charakteru. Mohlo by se zdát, že otázka úmrtí pojištěnce je nenáhodného charakteru, ale je potřebné zdůraznit, že okamžik úmrtí má náhodný charakter. S tím je spojena i otázka očekávané délky života, což je jedna z důležitých demografických charakteristik s celoplošným dopadem v oblasti sociálního zabezpečení. Stejně tak i v neživotním pojištění je otázka počtu škodových událostí řešena jako náhodná a rovněž i velikosti vzniklé škody se přisuzuje náhodný charakter.

 DOI: https://doi.org/10.11118/978-80-7509-898-6-0040

Pro jiné příklady můžeme odskočit do oblasti dopravy. I zde dochází ke mnoha událostem, jenž jsou obtížně předvídatelné. Každý se již zcela jistě setkal se zpožděním vlaků, avšak nikdo nedovede apriori předpovědět kdy k němu dojde a zejména v jakém rozsahu. Lze je předpovědět při výlukových pracích nebo nepříznivé předpovědi počasí, ale reálnou časovou prodlevu lze jen odhadovat na základě zkušeností z minulosti. Již vůbec nelze spolehlivě předpovědět kdy dojde k dopravním zácpám na dálnici v důsledku nehody a jak velké zdržení budou představovat.

Podobně jako v dopravě, i v provozu na počítačových sítích lze pozorovat procesy, které mají náhodný charakter. Může jít o návštěvnost jednotlivých webových stránek, počet došlých požadavků na server za určitou časovou jednotku a pod. Samozřejmostí je dnes i personalizace reklamy v prostředí internetu, které se rovněž opírá o statistické analýzy navštěvovaných stránek či obsahu vyhledávání ve vyhledávacích službách.

Nelze opomenout ani oblast automatizace výrobních procesů. Délka trvání jednotlivých technologických procesů je nezbytným prvkem pro úspěšné řešení automatizace. Zrovna tak životnost výrobků a jejich poruchovost jsou předmětem statistických analýz a statistické kontroly jakosti.

Z humanitních oborů uveďme alespoň oblast medicíny, kde právě medicínské testy a hodnocení účinnosti léčiv se opírají o statistické metody. Konec-konců i mnohé diagnostické metody lze interpretovat jako aplikovanou statistiku. Klinická psychologie je rovněž založena na statistických analýzách a dokonce se v mnoha případech sama stala základem pro rozvoj statistiky jako takové. Bez statistiky se neobejde ani tak zdánlivě vzdálená disciplína jako je politologie. Dnes tolik populární průzkumy volebních preferencí nejsou ničím jiným, než výsledkem statistického šetření.

Úhrnem lze tedy konstatovat, že všude tam, kde je nějaká věda patří statistika. To ale klade požadavky na vzdělávání ve všech vědních odborech. Porozumění a zejména schopnost správné interpretace výsledků statistického šetření by měly patřit do profesní výbavy každého vysokoškolsky vzdělaného odborníka.

2 Obsah kurzu

Z předchozího úvodu plyne, že kurz základů statistiky by měl být zařazen do převážné většiny studijních programů na vysokých školách. V rámci řešeného mezinárodního projektu „Innovative Open Source courses for Computer Science curriculum“ se při tom zaměřujeme zejména na podporu výuky otevřenými softwarovými prostředky.

Samotná obsahová náplň předmětu nikterak nevybočuje z běžně vyučovaných kurzů statistiky. V úvodních lekcích se posluchači seznamují se základy teorie pravděpodobnosti. Cílem této části je zejména seznámit účastníky kurzu s rozděleními pravděpo-

dobnosti používanými při intervalových odhadech, statistických testech hypotéz a se zákony velkých čísel. Svým rozsahem se shoduje se zahraničními učebnicemi [7],[2] nebo s domácími učebnicemi [3], [6] a [4].

Druhá část kurzu je věnována samotné matematické statistice. V teoretické části představuje výběrové charakteristiky, základy teorie odhadu, parametrické a neparametrické testy hypotéz a základy korelační a regresní analýzy.

Praktická část připravovaného kurzu je pak zaměřena na seznámení s prvky programovacího prostředí jazyka R. V rámci laboratorních cvičení se absolventi nejprve seznamují s prvky jazyka R, zejména s implementovanými datovými typy a strukturami dat. Jako podporu pro výuku pravděpodobnosti se seznámí s vybranými diskrétními a spojitými rozděleními pravděpodobnosti, pro něž jsou v jazyce R implementovány distribuční funkce, hustoty, kvantilové funkce a také generátory náhodných hodnot z daného rozdělení.

Důležitým prvkem prostředí R je bohatá škála nástrojů pro vizualizaci dat. Proto je této části věnována mimořádná a rozsáhlá pozornost. Posluchači se postupně naučí vytvářet i pokročilé nástroje grafické prezentace dat, jako jsou histogramy, sloupcové a koláčové grafy, box ploty, kvantilové grafy. Součástí této kapitoly jsou i bohaté možnosti formátování výsledných grafů.

Následně se studenti naučí práci se vstupními a výstupními soubory a vytváření vlastních funkcí. Tím se vytvářeny předpoklady pro jejich možnost samostatné práce s dostupnými reálnými daty, dostupnými z internetu. v rámci analýzy těchto dat se naučí využívat implementované nástroje pro testování statistických hypotéz, které si rovněž mohou vyzkoušet experimentováním s reálnými daty. V samotném závěru si pak ověří i jednoduchost korelační analýzy při použití implementovaných funkcí a různé formy regresních modelů.

Praktická část kurzu se opírá o literární zdroje [1] a [8]. Pro seznámení se samotným jazykem R pak slouží například učebnice [5] nebo [9].

3 Proč R

Jedním z charakteristických rysů statistického šetření a analýzy získaných dat je hromadnost. V dnešní době si lze jen stěží představit jejich zpracování bez využití výpočetní techniky. Je proto přirozené požadovat, aby každý kurz statistiky byl organicky propojen s využitím vhodných softwarových nástrojů. Při řešení otázky, jaké nástroje použít máme na výběr vícero možností.

Jednou alternativou je spolehnout se na kancelářské balíky a tabulkové výpočty. Při tom se obvykle vychází z teze, že ten MS Excell každý už jaksi zná a tak zvládne jakékoliv výpočty. Avšak při práci s rozsáhlejšími soubory dat se už používání tohoto typu aplikací ukazuje jako nepraktické a mnohdy je daný objem dat nezvládnutelný.

Jinou volbou mohou být komerční nástroje pro řešení statistických úloh, jako je například *Statistica* nebo programovací jazyk S-plus. Jde sice o výkonné nástroje, tomu ale zodpovídá i jejich cena.

Nabízí se i třetí varianta a to použít svobodné otevřené nástroje. proto jsme se rozhodli pro programovací jazyk R, jenž lze označit za implementaci programovacího jazyka S pod svobodnou licencí. Tento jazyk svojí popularitou a počtem uživatelů již předstihl komerční S a stalo se faktickým standardem v řadě oblastí statistiky. Každý si ho může zdarma nainstalovat z archivu na adrese `https://cran.r-project.org`. Zde je dostupný pro všechny běžné platformy operačních systémů. Navíc, v případě zájmu lze k němu instalovat i GUI R-studio.

Samozřejmě, bezplatnost není jeho jedinou předností. Ty které nás nejvíce ovlivnily při výběru tohoto nástroje lze shrnout do několika jednoduchých bodů:

- je bezplatný, většina platforem statistického softwaru stojí tisíce dolarů,
- k programu je dostupné velké množství rozšiřujících balíčků,
- R dokáže snadno importovat údaje z různých zdrojů,
- R má implementovaných mnoho pokročilých statistických nástrojů,
- R poskytuje interaktivní platformu na analýzu údajů,
- prostředí R nabízí vizualizaci dat v podobě vysoce kvalitních a estetických grafů,
- je nezávislé na platformě, kompatibilní s většinou nejrozšířenějších operačních systémů,
- je kompatibilní s programovacími jazyky jako C,C++, Python, Java.

Nelze opomenout ani skutečnost, že student získává další programátorskou zručnost a základní znalost jazyka specializovaného na statistické výpočty. Navíc jeho použití není vázáno na zakoupení žádné licence, takže i v budoucnu ho lze bez obtíží používat resp. nainstalovat na jakýkoliv počítač. To přináší jistý typ svobody, jenž při vazbě na konkrétní komerční nástroj absentuje.

4 Ukázka práce v prostředí R

Jak už bylo zmíněno, programovací prostředí R poskytuje možnost interaktivní práce. Proto uvedeme ukázku jeho použití po jednotlivých příkazech. Tyto příkazy lze ale rovněž uložit do souboru s příponou `.R` a následně spustit jako skript jazyka R.

V současnosti je aktuální vizualizace a analyzování různých dat souvisejících se šířením onemocnění COVID-19. Ani my nebudeme v tomto směru výjimkou a ilustrujeme některé prvky jazyka R právě na těchto datech. Nejprve si načteme data o pandemii v rámci SR ze sídla, kde jsou uloženy ve formátu `.csv`. Toho dosáhneme pomocí následující funkce:

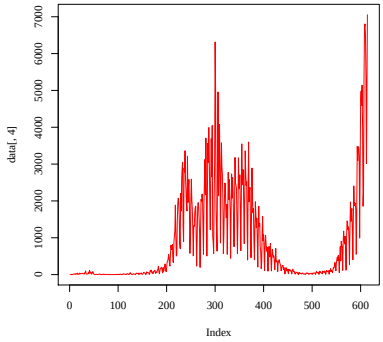
```
data<-read.csv("https://mapa.covid.chat/export/csv",header=T,sep=";")
```

Jejími argumenty jsou adresa vzdáleného zdroje dat, a informace o existenci záhlaví v datovém souboru a oddělovači hodnot, jímž je v tomto případě středník. O názvech jednotlivých položek struktury `data frame` se můžeme přesvědčit zavoláním funkce `names(data)`. Tak se dozvíme, že jednotlivé sloupce obsahují datum, počty potvrzených případů, počet vykonaných PCR testů, denní přírůstky a počet úmrtí. Chceme-li si jednoduše graficky prezentovat vývoj denních přírůstků, stačí zavolat funkci `plot()`. Jejími argumenty budou: čtvrtý sloupec struktury `data` a další argumenty pak upravují barevnost a typ zobrazené křivky v grafu.

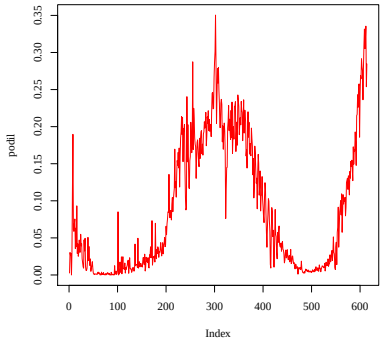
```
plot(data[,4],type="l",col="red")
```

Výsledek vidíme na obrázku 1. Absolutní velikost denních přírůstků však nemá zcela zásadní výpovědní hodnotu, protože počet případů je zcela jistě podmíněn počtem provedených testů. proto je lépe zobrazovat podíl pozitivních případů na provedených testech. ten si ovšem musíme nejprve přepočítat a až po té zobrazit. Výhodou v prostředí R je vektorová implementace funkcí, jež se vykonávají po složkách, což zjednodušuje zápis operace, jak ilustruje následující kód a jeho grafický výstup na obrázku 2.

```
podil<-data[,4]/data[,3]
plot(podil,type="l",col="red")
```



Obr. 1: *Graf přírůstků pozitivních PCR testů.*



Obr. 2: *Graf procentních podílů pozitivních na celkovém počtu testů.*

Tyto grafy však mají jisté nedostatky. Především si všimněme popisu jednotlivých os, kde bychom rádi doplnili informaci o údajích, jež graf prezentuje. Dále bychom jako značky na ose `x` asi rádi viděli datum. Toho lze dosáhnout potlačením vykreslení os ve funkci `plot()` nastavením na logickou hodnotu `FALSE`. Pomocí argumentů `xlab` a `ylab` definujeme popis jednotlivých os. Osy samotné pak vykreslíme pomocí funkce `axis()`.

Pro ukázku jsme z celého datasetu vybrali určitý fragment, který reprezentuje období, kdy na Slovensku proběhlo celoplošné testování a dodnes se vedou spory o tom, zda došlo k zastavení šíření infekce nebo naopak k navýšení pozitivních případů. Pro tento účel jsme vybrali úsek krátce před termínem testování, ke kterému došlo 1. 11. 2020 a druhé kolo 8. 11. 2020. Vývoj pak sledujeme až do konce roku 2020. Tyto údaje se nacházejí v data framu `data` na pozicích 230–300, jež musíme vyselektovat, např. definováním proměnné `pozice`, což ilustruje následující kód.

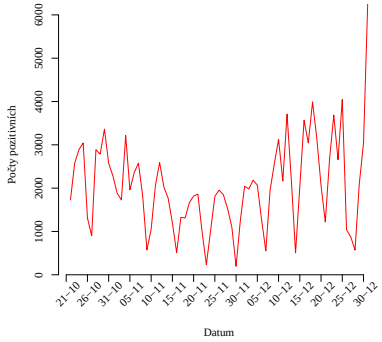
```
pozice<-seq(from=230,to=300,by=5)
plot(data[230:300,4],type="l",col="red",axes=FALSE,xlab="Datum",
ylab="Počty pozitivních",ylim=c(0,6000))
axis(2,at=seq(from=0,to=6000,by=1000),labels=seq(from=0,to=6000,by=1000))
axis(3,at=seq(from=0,to=70,by=5),labels=FALSE,pos=0)
```

Pro popis horizontální osy musíme nejdříve vytvořit příslušné popisy. Tyto datumy jsme uložili do proměnné `lablist`, přičemž jsme odstranili zbytečně dlouhý údaj o letopočtu, jenž je ve všech případech stejný, tady 2020. Popis pak přidáme k ose `x` pomocí funkce `text()`, která slouží k umístění textu na libovolnou pozici na kreslící ploše. Pomocí argumentu `srt` pak lze upravit i sklon textu. Výsledek vidíme na obrázku 3.

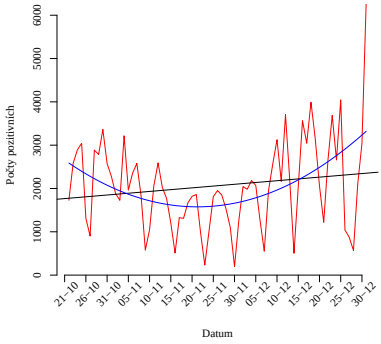
```
lablist<-substr(data[pozice,1],1,5)
text(x=seq(from=0,to=70,by=5),par("usr")[3] - 0.2,labels=lablist,srt=45,
pos=1,xpd = TRUE)
```

```
dny<-seq(0,70)
```

Důležitým nástrojem ve statistice je regresní analýza, jež se v základním kurzu vyučuje jen v rozsahu lineární regrese. Ta je v prostředí R implementována pomocí funkce `lm()`, což velmi usnadňuje její provedení. příslušnou regresní přímku pak lze v hotovém grafu přidat pomocí funkce `abline()`. Pokud bychom chtěli do regrese zařadit i vyšší mocniny, požijeme funkce `poly()`, kde v našem případě hodnota 2 udává, že jde o vyrovnání parabolou. Výslednou křivku pak zviditelníme pomocí funkce `curve()` s nastavením hodnoty parametru `add=T`, což opět ovlivní dokreslení křivky do již existujícího grafu.



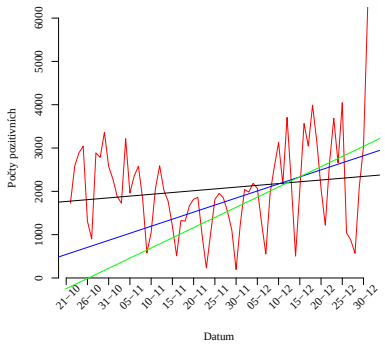
Obr. 3: *Graf podílů pozitivních PCR testů ve vybraném období roku 2020.*



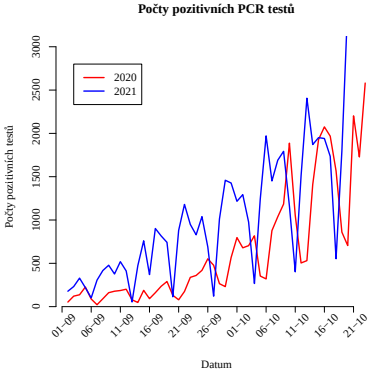
Obr. 4: *Graf podílů pozitivních PCR testů ve vybraném období roku 2020 s vyrovnáním přímkou a parabolou.*

```
rust_pocty<-lm(data[230:300,4]~dny)
abline(rust_pocty)
fit <- lm(data[230:300,4]~poly(dny,2,raw=TRUE))
curve(predict(fit,newdata=data.frame(dny=x)),add=T,col="blue")
```

Výsledný graf po přidání výsledných regresních křivek pak vidíme na obrázku 4. I když je z grafu zřejmé, že se nejedná o lineární vztah, nýbrž jsou zde výrazná sezónní minima víkendových dnů a nárůsty bezprostředně po nich, zejména parabola naznačuje výrazný nárůst pozitivních testů po uplynutí inkubační doby po termínech celoplošného testování.



Obr. 5: *Graf podílů pozitivních PCR testů ve sledovaném období se zobrazením trendů za několik období po plošném testování.*



Obr. 6: *Porovnání počtů pozitivních PCR testů ve stejných dnech v roce 2020 a 2021.*

Ještě lépe lze změny trendu v růstu počtu podílu pozitivních testů porovnat, pokud zobrazíme regresní přímky určené pro období před testováním (černá přímka na obrázku 6), a regresní přímky určené po uplynutí 10 dní od prvního testování (modrá přímka na obrázku 6) a 10 dní po druhém testování (zelená přímka na obrázku 6). Vývoj potvrzuje zrychlování v podílech pozitivních případů na celkovém počtu případů. Zmíněné regresní přímky přidáme do obrázku pomocí kódu

```
rust_podil<-lm(podil[250:300]~seq(20,70))
abline(rust_podil,col="blue")
rust_podil<-lm(podil[260:300]~seq(30,70))
abline(rust_podil,col="green")
```

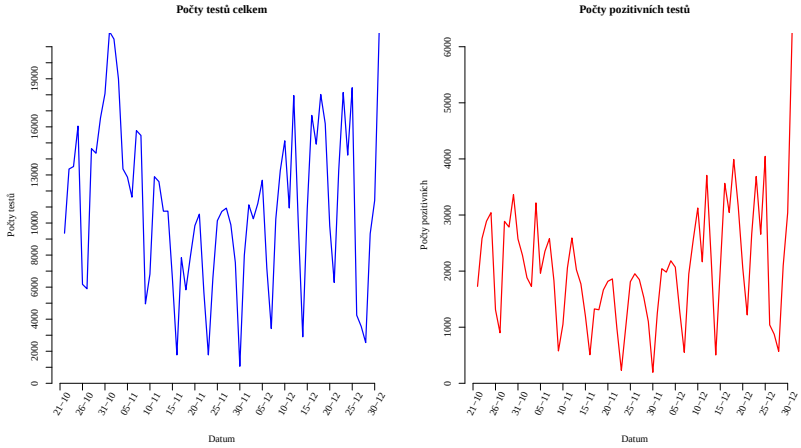
V současnosti je velice aktuální porovnávání vývoje letošní a loňské vlny. To lze provést zobrazením dvou křivek, odpovídajících stejným dnům v roce, do jednoho grafu. Nejprve definujeme proměnnou `pozice`, jež obsahuje údaje z období od 1.září 2020 do 21.října 2020. Do grafu zakreslíme příslušnou křivku červeně, přičemž uvedením proměnné `main` funkce `plot()` přidáme do grafu hlavní nadpis. K zobrazení údajů ze stejného období roku 2021 posuneme rozmezí indexů o 365 dní. Příslušnou křivku zobrazíme pomocí funkce `lines()`, čímž dojde k jejímu zobrazení ve stejném obrázku. K tomu požijeme následující kód:

```
pozice<-seq(from=180,to=231,by=5)
plot(data[180:231,4],type="l",col="red",lwd=2,axes=FALSE,xlab="Datum",
ylab="Počty pozitivních testů",ylim=c(0,3000),
main="Počty pozitivních PCR testů")
```

```
lines(data[545:593,4],type="l",col="blue",lwd=2)
axis(2,at=seq(from=0,to=3000,by=500),labels=seq(from=0,to=3000,by=500))
lablist<-substr(data[pozice,1],1,5)
axis(3,at=seq(from=0,to=50,by=5),labels=FALSE,tck=0.02,pos=0)
text(x=seq(from=0,to=50,by=5),par("usr")[3] - 0.2, labels = lablist,
srt = 45, pos = 1, xpd = TRUE)
```

Pro snazší orientaci ještě do grafu přidáme legendu pomocí funkce `legend()`, kde první dva parametry udávají pozici kde bude legenda umístěna. Konečný výsledek, jak je ilustrován na obrázku 6 dosáhneme přidáním kódu:

```
legend(2,2800, legend=c("2020", "2021"),col=c("red", "blue"), lty=1,lwd=2)
```



Obr. 7: *Graf počtu celkově provedených testů (vlevo) a počtu pozitivních případů (vpravo).*

Na závěr si ještě ilustrujme, jak lze porovnávat údaje tím, že je zobrazíme na dvou grafech vedle sebe, jak to vidíme na obrázku 4. Nejprve definujeme nové grafické zařízení a jeho rozměry. Vzhledem k tomu, že budeme chtít zobrazit dva grafy vedle sebe, zvolíme větší šířku grafického okna. Zadáme tedy kód:

```
dev.new(width=15, height=8, unit="in")
```

Použitím funkce `par()` a její proměnné `mfrow=c(1,2)` stanovíme, že výsledný obrázek bude mít jeden řádek a v něm dva sloupce, tedy dva grafy vedle sebe. Každé použití funkce `plot()` tedy vytvoří jeden nový graf. Celý kód pak vypadá takto:

```
dev.new(width=15, height=8, unit="in")
par(mfrow=c(1,2))
```

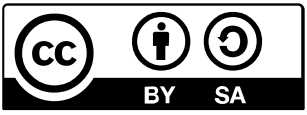
```
pozice<-seq(from=230,to=300,by=5)
plot(data[230:300,3],type="l",col="blue",lwd=2,axes=FALSE,xlab="Datum",
ylab="Počty testů",ylim=c(0,21000),main="Počty testů celkem")
axis(2,at=seq(from=0,to=21000,by=1000),labels=seq(from=0,to=21000,
by=1000))
lablist<-substr(data[pozice,1],1,5)
axis(3,at=seq(from=0,to=70,by=5),labels=FALSE,las=1,pos=0,tck=0.02)
text(x=seq(from=0,to=70,by=5),par("usr")[3] - 0.2, labels = lablist,
srt = 60, pos = 1, xpd = TRUE)
plot(data[230:300,4],type="l",col="red",lwd=2,axes=FALSE,xlab="Datum",
ylab="Počty pozitivních",ylim=c(0,6000),main="Počty pozitivních testů")
axis(2,at=seq(from=0,to=6000,by=1000),labels=seq(from=0,to=6000,
by=1000))
axis(3,at=seq(from=0,to=70,by=5),labels=FALSE,tck=0.02,pos=0)
text(x=seq(from=0,to=70,by=5),par("usr")[3] - 0.2, labels = lablist,
srt = 65, pos = 1, xpd = TRUE)
```

Literatura

1. Crawley, M. J. *Statistics: An Introduction Using R.* Addison-Wesley Publishing company, Boston, 2015.
2. DasGupta, A. *Fundamentals of Probability: A First Course*, New York, Springer-Verlag, 2010.
3. Holický, M. *Aplikace teorie pravděpodobnosti a matematické statistiky*, Praha, ČVUT, 2015.
4. Jakubowski, J., Sztencel, R. *Wstęp do teorii prawdopodobieństwa*, Warszawa, Script, 2010.
5. Kabacoff, R. *R in Action*, New York, Manning Publications, 2015.
6. Nánásiová, O., Kohnová, S.*Štatistika a pravdepodobnosť. Základy matematickej štatistiky a teórie pravdepodobnosti*, Bratislava, STU 2016.
7. Ross, S. M. *A first course in probability*, 10-th edition, Boston, Pearson, 2018.
8. Verzani, J. *Using R for Introductory Statistics*. Second edition, Boca Raton,CRC Press, Taylor & Francis Group, 2014.
9. Wickham, H., Grolemund, G. *R for Data Science*, Sebastopol, United States, O'Reilly Media, Inc, 2017.

Autor

RNDr. Aleš Kozubík, PhD., Katedra matematických metód a operačnej analýzy, Fakulta riadenia a informatiky, Žilinská univerzita v Žiline, Vysokoškolákov 8215/1, 010 26 Žilina, Slovenská republika, e-mail: alesko@frcatel.fri.uniza.sk



Matematická analýza podporovaná wxMaxima

Rudolf Blaško

1 Úvod do wxMaxima

wxMaxima je dialogové rozhraní pro systém počítačové algebry Maxima. wxMaxima nabízí menu a dialogová okna pro běžné příkazy, automatické dokončování, vložené grafy a jednoduché animace. wxMaxima je distribuován pod licencí GPL. Maxima patří mezi Open Source programy s otevřeným zdrojovým kódem. Program je možné kompilovat v různých OS, včetně Windows, GNU/Linuxu a MacOS X. Předkompilovanými program pro GNU/Linux a Windows lze bezplatně získat na stránce SourceForge `https://sourceforge.net/projects/maxima/files/`. Po spuštění prostředí wxMaxima se na obrazovce objeví okno s menu v horní části. Pod menu se nachází prostor, kde můžeme zadávat příkazy a kde se objevují výstupy.

```
(%i1)  First input line.
(%o1)  First output line.
(%i2)  Second input line.
(%o2)  Second output line.
```

Příkazy zadáváme na samostatné řádky (vstupní řádky), jejich provedení se zajistí současným stiskem kláves `Shift` a `Enter` nebo kliknutím v menu na ikonu (Send the current cell to maxima). Vstupní řádky jsou uvedeny oznámením (%i1) a výstupní řádky jsou uvedeny oznámením (%o1). Čísla pro vstupní a k němu odpovídající výstupní řádek jsou identické a na základě tohoto čísla se můžeme na obsah těchto řádků odvolávat.

```
(%i1)  solve(0=x+2,x);
(%o1)  [x = -2]
(%i2)  %i1;
(%o2)  solve(0 = x + 2, x)
(%i3)  %o1;
(%o3)  [x = -2]
```


DOI: https://doi.org/10.11118/978-80-7509-898-6-0050

Příkazy se provedou na nové samostatmé řádky (výstupní řádky). Příkazy na vstupních řádcích můžeme ukončit symbolem `;` (který systém automaticky doplní) nebo symbolem `$`, který potlačí zobrazení příslušného výstupu. Na vstupní řádek můžeme zadat i více příkazů, ale musíme je oddělit symboly `;` nebo `$`. Příkaz můžeme také strukturovat na více vstupních řádků.

```
(%i1)  a:2;b:3;solve(a*x+b*x^2=0,x)
(a)    2
(b)    3
(%o1)  [x = -2/3, x = 0]
(%i2)  a:2$ b:3$ solve(a*x+b*x^2=0,x);
(%o2)  [x = -2/3, x = 0]
(%i3)  a:2$
       b:3$
       solve(a*x+b*x^2=0,x);
(%o3)  [x = -2/3, x = 0]
```

Výstup můžeme uložit v různých tvarech a následně použít v jiných programech (LaTeX, editor rovnic MSWord, ...). Výstup (%o3) z předchozího okna můžeme:

- kopírovat (`CRL C` a `CRL V`), resp. kopírovat jako text (lze použít např. pro editor rovnic MSWord): `x=-2/3, x=0`,
- kopírovat jako LaTeX `\[x=-\frac{2}{3}\operatorname{,}x=0\]`,
- kopírovat jako MathML, obrázek, RTF, SVG...

Prostředí wxMaxima má dobře propracovaný help pro uživatele, který najdeme v menu `Help`. Help otevřeme také stisknutím klávesy `F1`. Manuál najdeme také na webové stránce `https://maxima.sourceforge.io/docs/manual/maxima_369.html`.

1.1 Základní příkazy

Příkazem `apropos` můžeme zjistit přesný název příkazu pomocí části jeho názvu.

```
(%i1)  apropos("plot")
(%o1)  [barsplot,boxplot,contour_plot,get_plot_option,gnuplot,...
```

Příkaz `describe` vypíše popis zadaného příkazu.

```
(%i1)  describe(plot2d);
       - -Function: plot2d
          plot2d (<expr><, <range_x><, <options><)
          plot2d (<expr_<>=<expr_<>, <range_x><, <range_y><, <options><)
          ...
(%o1)  true
```

Výrazy se zadávají pomocí obvyklých znaků operací, relací a funkcí. Argumenty funkcí a příkazů jsou v kulatých závorkách, symbol násobení `*` se musí zadat! Umocnění se zadává znakem `^` nebo dvojicí `**`.

Symbol `:` slouží k přiřazení hodnoty napravo výrazu na nalevo.

```
(%i1)  a:2$ b:3$ solve(a*x+b*x^2=0,x);
(%o1)  [x = -2/3, x = 0]
```

V menu `View` a podmenu `Display Equations` můžeme změnit zobrazení výstupních řádků na tvary `in 2D`, `as 1D ASCII` nebo `as ASCII Art`. Implicitně je nastaveno `in 2D`. Nastavení výstupu můžeme změnit i příkazem `set_display`. Nastavení na tvar `in 2D` má argument `none`.

```
(%i1)  x/sqrt(x^2+1);set_display('none)$

(%o1)  x/sqrt(x^2+1)              /* in 2D */
```

Pomocí argumentu `ascii` v příkazu `set_display` změníme výstup na tvar `as 1D ASCII` a pomocí argumentu `xml` na tvar `as ASCII Art`.

```
(%i1)  x/sqrt(x^2+1);set_display('ascii)$
(%o1)  x/sqrt(x^2+1)              /* as 1D ASCII */
(%i2)  x/sqrt(x^2+1);set_display('xml)$
                 x
(%o2)  ------------               /* as ASCII Art */
                2
       sqrt(x +1)
```

Příkazem `kill` můžeme odstranit proměnné se všemi jejich parametry a vlastnostmi z paměti.

```
(%i1)  kill(a,b)          /* removes all bindings from the arguments a,b */
(%i2)  kill(all)          /* removes all items on all infolists */
```

1.2 Práce s čísly a základní konstanty

Maxima může pracovat s reálnými čísly zapsanými v numerickém nebo symbolickém tvaru. Způsob zápisu reálných čísel můžeme nastavit v menu `Numeric` pomocí přepínače `Numeric Output` mezi numerickým a symbolickým zobrazováním. Také zde můžeme zvolit způsob a presnosnosť numerického zobrazování. O způsobu zobrazování rozhoduje nastavení proměnné `numer`.

Standardně se zobrazuje 16 číslic (včetně desetinné tečky). Přesnost zobrazení definuje proměnná `fpproc` a ovlivňuje zobrazení pomocí `bfloat`. Výstup `float` zobrazuje vždy stejně. Přesnost můžeme prakticky neomezeně zvýšit nebo snížit. Můžeme ji změnit globálně a také lokálně pouze pro jednu přeměnu nebo příkaz.

```
(%i1)  log(2);
(%o1)  log(2)
(%i2)  log(2),numer;
(%o2)  0.6931471805599453
(%i3)  float(log(2));
(%o3)  0.6931471805599453
(%i4)  bfloat(log(2));
(%o4)  6.931471805599453b-1
(%i5)  log(2),bfloat;
(%o5)  6.931471805599453b-1
(%i6)  bfloat(log(2)),fpprec=34;
(%o6)  6.9314718055994530941723212145817666b-1
(%i6)  bfloat(log(2)),fpprec=134;
(%o6)  6.9314718055994530941723212145[78digits]10205706857336855202357581 3b-1
```

Číselné konstanty e, π, i (imaginární jednotka) mají prefix `%`, tj. `%e`, `%pi`,`%i`. To platí i pro konstanty, které jsou součástí nebo výsledkem výpočtů. Také mají prefix `%`.

Maxima má předdefinované konstanty `inf`, `minf` pro reálné nekonečná ∞, $-\infty$ a `infinity` pro komplexní nekonečno.

Logické konstanty `true` a `false` představují pravdu a nepravdu.

```
(%i1)  %pi; %i; %e;
(%o1)  π %i %e
(%i2)  minf; inf;
(%o2)  −∞ ∞
(%i3)  infinity;
(%o3)  infinity
```

Komplexním číslům se v tomto kurzu nevěnujeme, proto pouze zmíníme jak se zobrazují. Komplexní čísla se implicitně zadávají algebraickém tvaru (`rectform`). Do goniometrického (exponenciálního) tvaru je můžeme převést pomocí příkazu `polarform`.

```
(%i1)  z:1+%i;
(z)    i+1
(%i2)  polarform(z)+rectform(z);
(%o2)  √2 e^(i π/4) +i+1
```

1.3 Přiřazení a funkce

Operátor `:` používáme na přiřazení hodnot nebo výrazů proměnným. Funkce definujeme pomocí přiřazení `:=`.

```
(%i1)  f(x):=x^2+2*x+3;
(%o1)  f(x) := x^2 + 2 * x + 3
(%i6)  f(x);f(y);f(x+1);f(-2);f(1);
(%o2)  x^2 + 2 * x + 3
(%o3)  y^2 + 2 * y + 3
(%o4)  (x + 1)^2 + 2 * (x + 1) + 3
(%o5)  3
(%o6)  6
```

Maxima obsahuje mnohem více funkcí než standardní programovací jazyky. Jsou to nejen samotné reálné funkce, ale také různé funkce pro jejich podporu. Mezi základní funkce patří `sign(x)`, `abs(x)`, `floor(x)` (dolní celá část čísla x), `round(x)` (zaokrouhlí x na nejbližší celé číslo), `truncate(x)` (odstraní všechny číslice za desetinnou tečkou), `ceiling(x)` (horní celá část čísla x).

```
(%i2)  f(x):=sign(x)$ print(f(-3.2),f(0),f(3.2))$
       neg zero pos
(%i4)  f(x):=abs(x)$ print(f(-3.2),f(0),f(3.2))$
       3.2 0 3.2
(%i6)  f(x):=floor(x)$ print(f(-3.6),f(-3.2),f(-1),f(0),f(1),f(3.2),f(3.6))$
       -4 -4 -1 0 1 3 3
(%i8)  f(x):=round(x)$ print(f(-3.6),f(-3.2),f(-1),f(0),f(1),f(3.2),f(3.6))$
       -4 -3 -1 0 1 3 4
(%i10) f(x):=truncate(x)$ print(f(-3.6),f(-3.2),f(-1),f(0),f(1),f(3.2),f(3.6))$
       -3 -3 -1 0 1 3 3
(%i12) f(x):=ceiling(x)$ print(f(-3.6),f(-3.2),f(-1),f(0),f(1),f(3.2),f(3.6))$
       -3 -3 -1 0 1 4 4
```

Pro formátování výpisu jsme použili příkaz `print`.

```
(%i3)  a:2$ b:log(2),numer$ print("Logarithm of a-number",a," is ",log(a),"=",b)$
       Logarithm of a number 2 is log (2) = 0.6931471805599453
```

Maxima obsahuje mnoho elementárních funkcí. Jsou to například `exp(x)=%e^x`, `log(x)`, goniometrické funkce a k nim inverzní funkce `sin(x)`, `cos(x)`, `tan(x)`, `cot(x)`, `asin(x)`, `acos(x)`, `atan(x)`, `acot(x)`, hyperbolické a k nim inverzní funkce `sinh(x)`, `cosh(x)`, `tanh(x)`, `coth(x)` `asinh(x)`, `acosh(x)`, `atanh(x)`, `acoth(x)` atd.

Maxima obsahuje také mnoho funkcí pro jejich podporu. Některé z nich nejsou implementovány přímo v prostředí wxMaxima, ale v externích knihovnách nazývaných balíčky (packages). Tyto balíčky se do systému načítají pomocí příkazu `load`. Na ukázku uvedeme balíček `spangle` pro podporu práce s goniometrickými funkcemi.

```
(%i2)  print(tan(%pi/8),ratsimp(tan(%pi/8)),trigsimp(tan(%pi/8)))$

       tan(π/8)  tan(π/8)  sin(π/8)/cos(π/8)

(%i3)  load(spangl);

(%o3)  ../share/trigonometry/spangl.mac

(%i4)  tan(%pi/8);

(%o4)  √2 − 1
```

1.4 Práce s výrazy

Operace a výpočty programu Maxima probíhají v nějakém prostředí, ve kterém systém předpokládá platnost určitých podmínek. Tyto podmínky můžeme měnit. Mnohokrát potřebujeme změnit podmínky pouze lokálně pro nějaký konkrétní výpočet aniž

abychom měnili globální nastavení. K tomuto účelu posytuje Maxima velmi účinný příkaz `ev`, který umožňuje definovat specifické prostředí v rámci jednoho příkazu.

Po zadání příkazu `ev(a,b1,b2,...,bn)` se vyhodnotí výraz `a` při splnění podmínek `b1`, `b2`, ..., `bn`. Tyto podmínky mohou být rovnice, přiřazení, funkce, přepínače (logické nastavení). Na ukázku uvádíme příklad řešení kvadratické rovnice pomocí příkazu `solve`. Proměnné `a`, `b`, `c` po provedení příkazu `ev` nemají přiřazené hodnoty.

```
(%i1)  ev(solve(a*x^2+b*x+c=0,x),a:2,b:-1,c=-3);
(%o1)  [x = 3/2, x = -1]
(%i2)  solve(a*x^2+b*x+c=0,x);
(%o2)  [x = -(sqrt(b^2-4*a*c)+b)/(2*a), x = (sqrt(b^2-4*a*c)-b)/(2*a)]
```

Na zjednodušení a úpravy různých výrazů nabízí Maxima několik příkazů. Základní funkce najdeme v menu `Simplify`. S příkazy `ratsimp` a `trigsimp` jsme se již setkali a při úpravě hodnoty `tan(%pi/8)` neměli žádoucí efekt.

Maxima nabízí pomocí příkazu `example` příklady k jednotlivým příkazům. Podívejme se na některé ukázky, které nabízí `example(ratsimp)`.

```
(%i2)  f(x):=b*(a/b-x)+b*x+a$ print(f(x),"?",ratsimp(f(x)))$
       bx + b(a/b - x) + a ? 2a
(%i3)  ratsimp(a+1/a);
(%o3)  (a^2+1)/a
(%i4)  ev(x^(a+1/a),ratsimp);
(%o4)  x^(a+1/a)
(%i5)  ev(x^(a+1/a),ratsimpexpons);
(%o5)  x^((a^2+1)/a)
```

Funkce `expand` roznásobí příslušné členy ve výrazu. Funkce `factor` daný výraz naopak rozloží. Funkce `gfactor` tak činí nad polem komplexních čísel.

```
(%i1)  f(x):=(x+1)*(x^2-4)*(x^2+4)$
(%i3)  ratsimp(f(x));expand(f(x));
(%o2)  x^5 + x^4 - 16*x - 16
(%o3)  x^5 + x^4 - 16*x - 16
(%i6)  factor(f(x));gfactor(f(x));factor(100);
```

(%o4) $(x-2)(x+1)(x+2)(x^2+4)$
(%o5) $(x-2)(x+1)(x+2)(x-2\%\mathrm{i})(x+2\%\mathrm{i})$
(%o6) $2^2 5^2$

Racionální lomenou funkci rozložíme na parciální zlomky pomocí příkazu `partfrac`.

```
(%i1) partfrac((x+1)/(x^2-2*x+1),x);
```
(%o1) $\frac{1}{x-1}+\frac{2}{(x-1)^2}$

Substituovat výrazy můžeme pomocí příkazů `subst(a,b,c)` a `ratsubst(a,b,c)`. Výraz `a` bude nahrazen za výraz `b` a následně dosazen do výrazu `c`. Při použití příkazu `subst` musí být `b` nejjednodušší částí (atomem) nebo kompletním podvýrazom výrazu `c`. V příkladu není podvýraz `x+y` kompletní (chybí `z`). Příkaz `ratsubst` výsledný výraz i upraví.

```
(%i2) subst(x+y,a,a^2+b^2);ratsubst(x+y,a,a^2+b^2);
```
(%o1) $(y+x)^2+b^2$
(%o2) $y^2+2xy+x^2+b^2$
```
(%i4) subst(a,x+y,x+y+z);ratsubst(a,x+y,x+y+z);
```
(%o3) $z+y+x$
(%o4) $z+a$

1.5 Limity a derivování

V menu `Calulus` najdeme funkce na řešení základních úloh matematické analýzy (limity, derivování, integrování, součty řad, rozklad funkce do Taylorova polynomu...).

Limity počítáme pomocí příkazu `limit`. Poslední parametr určuje směr jednostranných limit, má hodnoty `plus`, resp. `minus` a je nepovinný. Pokud není určen, Maxima počítá limitu jako komplexní. Příkazy `limit(f(x),x,a)`, `limit(f(x),x,a,plus)` vypočítáme limity $\lim_{x\to a} f(x)$, $\lim_{x\to a^+} f(x)$.

```
(%i4) limit(1/x,x,0);limit(1/x,x,0,plus);limit(1/x,x,0,minus);limit(1/x,t,0);
```
(%o1) $infinity$
(%o2) ∞
(%o3) $-\infty$
(%o4) $\frac{1}{x}$

Pokud použijeme před příkazem apostrof `'`, příkaz se neprovede, pouze zobrazí.

```
(%i2)  limit(((1-n)/(1+3*n))^(1+4*n),n,inf);'limit(((1-n)/(1+3*n))^(1+4*n),n,inf);
(%o1)  0
(%o2)  lim_{n→∞} ((1-n)/(3n+1))^(4n+1)
```

Derivace počítáme pomocí příkazu `diff`. Parametr, který určuje rád derivace je nepovinný.

```
(%i4)  f(x):=2*x^4-3*x+sin(x);
       print("f'=",diff(f(x),x),"=",diff(f(x),x,1))$
       print("f''=",diff(diff(f(x),x),x),"=",diff(f(x),x,2),"=",diff(f(x),x,1,x,1))$
       print("f^(10)=",diff(f(x),x,10),"=",diff(f(x),x,1,x,9))$
(%o1)  f(x) := 2x^4 - 3x + sin(x)
       f' = cos(x) + 8x^3 - 3 = cos(x) + 8x^3 - 3
       f'' = 24x^2 - sin(x) = 24x^2 - sin(x) = 24x^2 - sin(x)
       f_(10) = - sin(x) = - sin(x)
```

Parciální derivace počítáme pomocí stejného příkazu.

```
(%i3)  g(x,y):=x^3*y^2-1;
       print("g'_x=",diff(g(x,y),x),", resp. g'_y=",diff(g(x,y),y,1))$
       print("g''_(xx)=",diff(g(x,y),x,2),", resp. g''_(yx)=",diff(g(x,y),y,1,x,1))$
(%o1)  g(x,y):=x^3*y^2-1
       g'_x = 3x^2 y^2, resp. g'_y = 2x^3 y
       g''_(xx) = 6xy^2, resp. g''_(yx) = 6x^2 y
```

Taylorův polynom n-tého stupně vypočítáme pomocí příkazu `taylor`. Tento příkaz najdeme v menu `Calculus` a podmenu`Get Series...`. Taylorova řada funkce f stupně n v středu c vypočítáme příkazem `taylor(f(x),x,c,n)`. Jeho koeficienty dostaneme použitím příkazu `coeff`. Použití tohoto příkazu je závislé na příkazu `taylor`.

```
(%i1)  t1:taylor(sin(x),x,0,5); t2:taylor(sin(x),x,-1,5);
(t1)   x - x^3/6 + x^5/120 + ...
(t2)   - sin(1) + cos(1)(x+1) + sin(1)(x+1)^2/2 - cos(1)(x+1)^3/6 - sin(1)(x+1)^4/24 + cos(1)(x+1)^5/120 + ...
(%i3)  print(coeff(sin(x),x,5)," and ",coeff(t1,x,5)," and ",coeff(t2,x,5))$
       0 and 1/120 and cos(1)/120
```

Taylorův polynom polynomu je opět polynom, pouze je vyjádřen v jiném tvaru. Prakticky se změní pouze souřadnicový systém, ve kterém polynom vyjadřujeme. Začátek systému se posune z bodu 0 do bodu -1. V následujícím příkladu je Taylorův polynom daného polynomu vypočítaný i jiným způsobem. Příkaz `taylor` dává na konec tři tečky, i když je rozvoj uzavřen.

```
(%i1)  f(x):=2*x^5-x^4-3*x^3-x+1;
```
(%o1) $f(x) := 2x^5 - x^4 + (-3)x^3 - x + 1$
```
(%i2)  tp1:taylor(f(x),x,-1,5);
```
(tp1) $2 + 4(x+1) - 17(x+1)^2 + 21(x+1)^3 - 11(x+1)^4 + 2(x+1)^5 + \cdots$
```
(%i4)  ratsimp(tp1);expand(tp1);
```
(%o3) $2x^5 - x^4 - 3x^3 - x + 1$

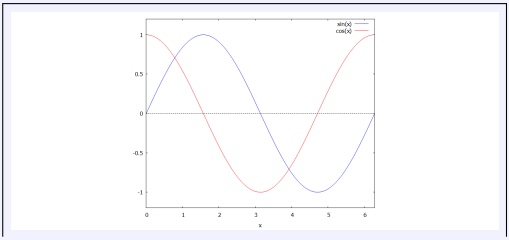
(%o4) $2x^5 - x^4 - 3x^3 - x + 1$
```
(%i6)  tpx:ratsubst(t,x+1,f(x));subst(x+1,t,tpx);
```
(tpx) $2t^5 - 11t^4 + 21t^3 - 17t^2 + 4t + 2$

(tp2) $2(x+1)^5 - 11(x+1)^4 + 21(x+1)^3 - 17(x+1)^2 + 4(x+1) + 2$
```
(%i7)  tp1-tp2;
```
(%o7) $0 + \cdots$

1.6 Grafy funkcí

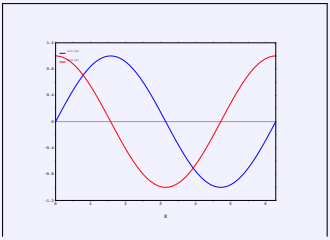
Graf funkce můžeme vykreslit několika způsoby. Nejjednodušší je zvolit v menu `Plot` podmenu `Plot 2d...`. Pokud zvolíme `Format=gnuplot`, funkci vykreslí příkaz `plot2d` pomocí Open Source programu `gnuplot` do nového okna. `gnuplot` se automaticky instaluje spolu s programem Maxima.

```
(%i1)  plot2d([sin(x),cos(x)],[x,-%pi,2*%pi],[y,-1.2,1.2],[plot_format, gnuplot])$
```



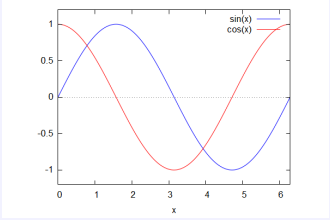
Pokud zvolíme `Format=wxmaxima`, Maxima vykreslí graf pomocí příkazu `plot2d` do nového okna. Obrázek můžeme uložit pouze do postscriptu.

```
(%i1)  plot2d([sin(x),cos(x)],[x,-%pi,2*%pi],[y,-1.2,1.2],[plot_format, xmaxima])$
```



Pokud zvolíme `Format=inline`, Maxima vykreslí graf pomocí příkazu `wxplot2d` do svého prostředí.

```
(%i1)  wxplot2d([sin(x),cos(x)],[x,-%pi,2*%pi],[y,-1.2,1.2])$
```

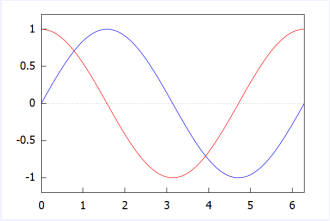


(%o1)

Příkazy `plot2d` a `wxplot2d` mají stejnou syntaxi a mají mnohem více parametrů. Parametry můžeme zjistit například příkazem `describe(plot2d)`.

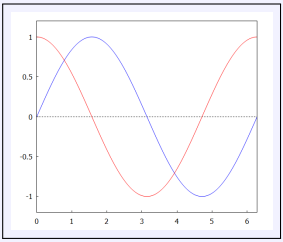
Na tisk grafu funkcí je výhodnější použít příkaz `wxdraw2d` nebo `draw2d`, který je vhodné směřovat na výstup programu `gnuplot`. Tyto příkazy mají trochu jinou syntaxi jako příkazy `wxplot2d`, resp. `plot2d`. Parametry tisku jsou v nich jednodušší a přehlednější. Vykreslována funkce musí být umístěna v příkazu `explicit`, `parametric` nebo `implicit`.

```
(%i1) wxdraw2d(xaxis=true,yaxis=true,xrange=[0,2*%pi],yrange=[-1.2,1.2],
      color=blue,explicit((sin(x)),x,0,2*%pi),
      color=red,explicit((cos(x)),x,0,2*%pi))$
```



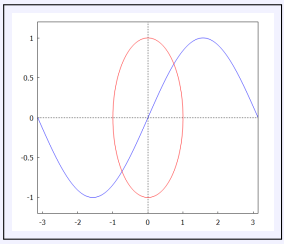
(%o1)

```
(%i1) draw2d(xaxis=true,yaxis=true,xrange=[0,2*%pi],yrange=[-1.2,1.2],
      color=blue,explicit((sin(x)),x,0,2*%pi),
      color=red,explicit((cos(x)),x,0,2*%pi))$
```



Parametrickou křivku nebo funkci vykreslíme podobným způsobem.

```
(%i1)  draw2d(xaxis=true,yaxis=true,xrange=[-%pi,%pi],yrange=[-1.2,1.2],
       color=blue,explicit((sin(x)),x,-%pi,%pi),
       color=red,nticks=300,parametric(cos(t),sin(t),t,0,2*%pi))$
```



2 Posloupnosti a řady

Posloupnosti můžeme v programu Maxima vytvořit například pomocí příkazu `makelist` nebo příkazy cyklu `for` - `do`.

Posloupnost (reálných čísel) je každá posloupnost $\{a_n\}_{n=1}^{\infty}$, jejíž členy jsou reálná čísla $a_n \in R$ (tj. zobrazení $N \to R$).

- **Explicitní zadání** (obecné vyjádření) člena a_n jako funkce proměnné n.
- **Rekurentní zadání** prvního člena a zadání a_n pomocí předchozích členů.

$\{a_n\}_{n=1}^{\infty} = \{2n-1\}_{n=1}^{\infty} = \{1, 3, 5, \ldots\}$.

- Explicitní zadání $a_n = 2n - 1$, $n \in N$.
- Rekurentní zadání $a_1 = 1$, $A_{n+1} = a_n + 2$, $n \in N$.

```
(%i3)  a(n):=2*n-1$ S:makelist(a(n),n,1,7);
(S)    [1, 3, 5, 7, 9, 11, 13]
(%i4)  an:1$ (for n:1 thru 7 do (print(an),an:an+2))$
       1
```

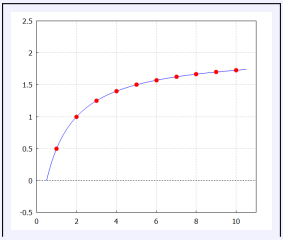
```
3
5
7
9
11
13
```

Příkaz `makelist` vytvoří seznam, který můžeme zobrazit i jako celek i po členech.

```
(%i2)  S1:makelist(2*n^2-1,n,1,10);S2:makelist(2*n^2-1,n,2,10,2);
(S1)   [1, 7, 17, 31, 49, 71, 97, 127, 161, 199]
(S2)   [7, 31, 71, 127, 199]
(%i4)  S1[1];S1[10];
(%o3)  1
(%o4)  199
```

Uspořádané dvojice se dávají do hranatých závorek a můžeme je zobrazovat jako body v rovině. V následujícím příkladu je vygenerována posloupnost i se svými vzory a následně vykreslena příkazem `draw2d`.

```
(%i1)  S1:makelist([n,(2*n-1)/(n+1)],n,1,10);
(S1)   [[1, 1/2], [2, 1], [3, 5/4], [4, 7/5], [5, 3/2], [6, 11/7], [7, 13/8], [8, 5/3], [9, 17/10], [10, 19/11]]
(%i2)  draw2d(grid=true,xaxis=true,yaxis=true,xrange=[0,11],yrange=[-0.5,2.5],
       color=blue,explicit((2*n-1)/(n+1),n,0.5,10.5),
       point_type=7,color=red,points(S1))$
```



Pomocí příkazů `for – do` vypíšeme několik členů posloupnosti $\left\{2n^2-1\right\}_{n=1}^{\infty}$.

```
(%i1)  (for n:1 thru 12 do (a_n: 2*n^2-1, print(a_n)) )$
       1
       7
       17
```

```
        31
        49
        71
        97
        127
        161
        199
        241
        287
```

Pěkným příkladem použití příkazů `for` – `do` je Fibonacciho posloupnost.

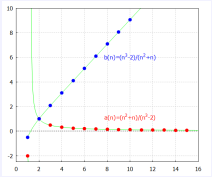
```
(%i3)   a0:0$ a1:1$ (for i:1 thru 12 do (an:a1+a0, print(an), a1:a0, a0:an))$
        1
        1
        2
        3
        5
        8
        13
        21
        34
        55
        89
        144
```

.....

- $\lim\limits_{n\to\infty} \frac{n^2+n}{n^3-2} = \lim\limits_{n\to\infty} \frac{n^3(n^{-1}+n^{-2})}{n^3(1-2n^{-3})} = \lim\limits_{n\to\infty} \frac{n^{-1}+n^{-2}}{1-2n^{-3}} = \frac{0+0}{1-0} = 0.$
- $\lim\limits_{n\to\infty} \frac{n^3-2}{n^2+n} = \lim\limits_{n\to\infty} \frac{n^2(n-2n^{-2})}{n^2(1+n^{-1})} = \lim\limits_{n\to\infty} \frac{n-2n^{-2}}{1+n^{-1}} = \frac{\infty-0}{1+0} = \infty.$

```
(%i1)   a(n):=(n^2+n)/(n^3-2)$ Sa:makelist([n,a(n)],n,1,15)$
        b(n):=(n^3-2)/(n^2+n)$ Sb:makelist([n,b(n)],n,1,15)$
        print("limit a(n)=",limit(a(n),n,inf),"  limit b(n)=",limit(b(n),n,inf))$
        draw2d(grid=true,xaxis=true,yaxis=true,xrange=[0,16],yrange=[-2.5,10],
        color=green,explicit(a(n),n,1,16),point_type=7,color=red,points(Sa),
          label(["a(n)=(n^2+n)/(n^3-2)",10,a(10)+1]),
        color=green,explicit(b(n),n,1,16),point_type=7,color=blue,points(Sb),
```

```
label(["b(n)=(n^3-2)/(n^2+n)",10,6]))$
```
(%o1) $limita(n) = 0 \quad limitb(n) = \infty$



Konečný i nekonečný součet vypočítáme pomocí příkazu `sum`.

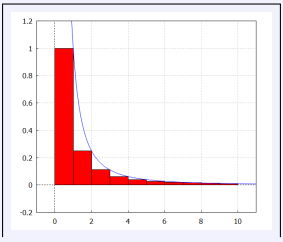
```
(%i1) sum(2*n^2-1,n,1,8);
```
(%o1) 400

Pomocí tohoto příkazu dokáže Maxima vypočítat přesný součet některých nekonečných řad. Součet řady můžeme zadat v menu `Calculus` a podmenu `Vypočítat Sum...`.

```
(%i2) sum(1/k^2,k,1,inf),simpsum; sum(1/k^2,k,1,inf);
```
(%o1) $\frac{\pi^2}{6}$

(%o2) $\sum_{k=1}^{\infty} \left(\frac{1}{k^2}\right)$

Číselná řada z předchozího příkladu může být graficky znázorněna následovně.

```
(%i1) a(n):=1/n^2$ rec:makelist(rectangle([i-1,0],[i,a(i)]),i,1,10)$
      draw2d(grid=true,xaxis=true,yaxis=true,xrange=[-1,11],yrange=[-0.2,1.2],
      border=true,color=black,fill_color=red,rec,
      color=blue,explicit(a(n),n,0,11))$
```


Číselné řady úzce souvisejí s posloupnostmi a zobecňují pojem sčítání na nekonečný počet sčítanců. Jednoduchým příkladem jsou zlomky a periodické čísla.

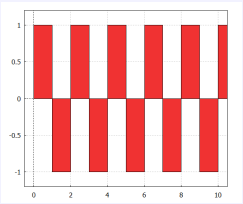
$\{a_n\}_{n=1}^{\infty}$ je posloupnost.

$$\Rightarrow \sum_{n=1}^{\infty} a_n = a_1 + a_2 + a_3 + \cdots + a_n + \cdots$$ se nazývá **(nekonečná číselná) řada**.

Pro nekonečné řady neplatí některá pravidla platná pro konečné počty sčítanců. Neplatí např. asociativní zákon:

$$\sum_{n=1}^{\infty} (-1)^{n+1} = \begin{cases} (1-1)+(1-1)+(1-1)+\cdots = 0+0+0+\cdots = 0, \\ 1+(-1+1)+(-1+1)+\cdots = 1+0+0+\cdots = 1. \end{cases}$$

```
(%i1) a(n):=(-1)^(n+1)$ rec:makelist(rectangle([i-1,0],[i,a(i)]),i,1,11)$
draw2d(grid=true,xaxis=true,yaxis=true,xrange=[-.5,10.5],yrange=[-1.2,1.2],
border=true,color=black,fill_color=red,rec)$
```



$\sum_{n=1}^{\infty} a_n$ je číselná řada.

- $s_k = \sum_{i=1}^{k} a_i = a_1 + a_2 + \cdots + a_k$, $k \in N$ se nazývá **k-tý částečný součet řady** $\sum_{n=1}^{\infty} a_n$.
- $r_k = \sum_{i=k+1}^{\infty} a_k = a_{k+1} + a_{k+2} + a_{k+3} + \cdots$ se nazývá **k-tý zbytek řady** $\sum_{n=1}^{\infty} a_n$.
- $\{s_k\}_{k=1}^{\infty} = \{s_n\}_{n=1}^{\infty}$ se nazývá **posloupnost částečných součtů řady** $\sum_{n=1}^{\infty} a_n$.

Vztah mezi $\sum_{n=1}^{\infty} a_n$ a posloupností $\{s_n\}_{n=1}^{\infty}$ je vzájemně jednoznačný.

Pro $\{s_n\}_{n=1}^{\infty}$ a $\sum_{n=1}^{\infty} a_n$ platí:

- $s_1 = a_1$.
- $a_1 = s_1 = s_1 - s_0$, kde $s_0 = 0$.
- $s_2 = a_1 + a_2 = s_1 + a_2$.
- $a_2 = s_2 - s_1$.
- $s_3 = a_1 + a_2 + a_3 = s_2 + a_3$.
- $a_3 = s_3 - s_2$.

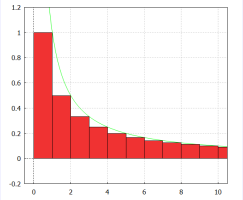
  ...
- $s_n = a_1 + a_2 + \cdots + a_{n-1} + a_n = s_{n-1} + a_n$.
- $a_n = s_n - s_{n-1}$, $n \in N$.

Součet řady $\sum_{n=1}^{\infty} a_n$ se nazývá $\lim_{n\to\infty} s_n = s \in R^*$ (pokud existuje), označení $\sum_{n=1}^{\infty} a_n = s$.

Harmonická řada

$\sum_{n=1}^{\infty} \frac{1}{n} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \cdots = \infty.$

```
(%i1) a(n):=1/n$ rec:makelist(rectangle([i-1,0],[i,a(i)]),i,1,11)$
draw2d(grid=true,xaxis=true,yaxis=true,xrange=[-.5,10.5],yrange=[-.2,1.2],
color=green,explicit(a(n),n,.5,11),
border=true,color=black,fill_color=light_red,rec)$
```

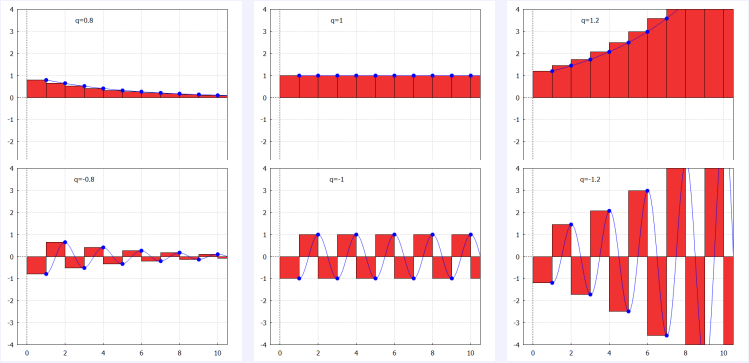


Geometrická řada

$\sum_{n=1}^{\infty} q^{n-1} = 1 + q + q^2 + \cdots = \frac{1}{1-q}$ pro všechny $q \in (-1; 1)$.

V následujícím příkladu stačí měnit na začátku hodnotu q.

```
(%i1) q:0.8$ a(n,q):=q^n$ peca:makelist([i,a(i,q)],i,1,11)$
reca:makelist(rectangle([i-1,0],[i,a(i,q)]),i,1,11)$
draw2d(grid=true,xaxis=true,yaxis=true,xrange=[-.5,10.5],yrange=[-4,4],
border=true,color=black,fill_color=light_red,reca,
label([concat("q=",string(q)),3,3.5]), color=blue,explicit(a(n,q),n,1,11),
point_type=7,color=blue,points(peca))$
```




```
(%i4) sq(q):=sum(q^n,n,1,inf)$
      sq(1/2),simpsum; sq(1/3),simpsum; sq(-1/2),simpsum; sq(2),simpsum;
(%i1) 1
(%i2) 1/2
(%i3) -1/3
(%i4) sum: sum is divergent.
```

3 Funkce

Funkce $y = f(x)$, $x \in D(f)$, tj. $f\colon D(f) \to H(f)$.

- Množina $\{[x;y] \in R^2; x \in D(f),\ y = f(x)\}$ se nazývá **graf funkce** f.
- **Funkce reálné proměnné**, jestliže definiční obor $D(f) \subset R$.
- **Reálná funkce**, pokud obor hodnot $H(f) \subset R$.

$y = f(x)$, $x \in A$ se nazývá:

- **Injektivní (injekce, prostá)**, jestliže pro všechny $x_1, x_2 \in A$, $x_1 \neq x_2$ platí $f(x_1) \neq f(x_2)$,

  tj. z rovnosti $f(x_1) = f(x_2)$ plyne rovnost $x_1 = x_2$.

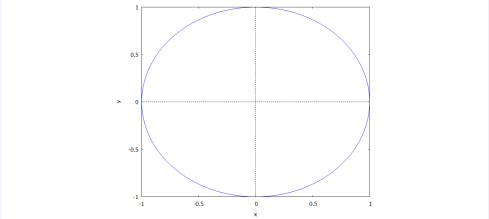
- **Surjektivní (surjekce, na množinu)**, pokud $f(A) = B$,
  tj. ke každému $y \in B$ existuje $x \in A$ takové, že $y = f(x)$.
- **Bijektivní (bijekce)**, pokud je injektivní a surjektivní.

$y = f(x)$, $x \in D(f)$ se vyjadřuje:

- **Explicitně**, tj. analyticky vzorcem $y = f(x)$, $x \in D(f)$.
- **Parametricky** rovnicemi $x = \varphi(t)$, $y = \psi(t)$, $t \in J$, $J \subset R$, kde $\varphi, \psi\colon J \to R$.
  Parametr t má pomocný význam.
- **Implicitně** rovnicí $f(x, y) = 0$, kde $f\colon R^2 \to R$ a podmínkami pro $[x; y]$.

Pokud chceme v programu Maxima zobrazit funkci zadanou implicitně, musíme načíst knihovnu `implicit_plot`.

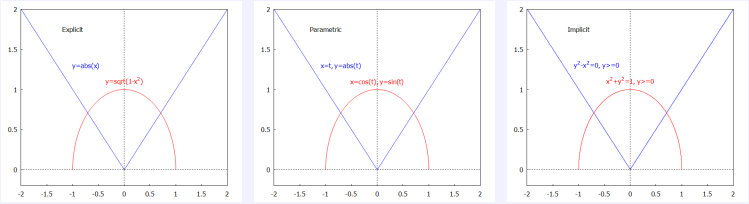
```
(%i1)  load(implicit_plot);
(%o1)  ../share/contrib/implicit_plot.lisp
(%i2)  implicit_plot(x^2+y^2-1,[x,-1,1],[y,-1,1])$
       implicit_plot is now obsolete. Using plot2d instead:
           plot2d (y^2+x^2-1=0,[x,-1,1],[y,-1,1])
(%i2)  plot2d(x^2+y^2-1=0,[x,-1,1],[y,-1,1])$     /* is correct */
```



Funkci f: $y = |x|$, $x \in R$ můžeme zadat např.:

- Explicitně: $y = \sqrt{x^2}$ resp. $y = \max\{-x, x\}$.
- Parametricky: $x = t$, $y = |t|$ $t \in R$, resp. $x = t$, $y = \sqrt{t^2}$, $t \in R$.
- Implicitně: $y^2 - x^2 = 0$, $y \geq 0$, resp. $y - |x| = 0$.

```
(%i1)  load(implicit_plot)$
(%i2)  draw2d(xaxis=true, yaxis=true, xrange=[-2,2], yrange=[-.2,2],
       color=blue, explicit(abs(x),x,-2,2), label(["y=abs(x)",-.75,1.3]),
       color=red, explicit(sqrt(1-x^2),x,-1,1), label(["y=sqrt(1-x^2)",0,1.1]),
       color=black, label(["Explicit",-1,1.75]))$
(%i3)  draw2d(xaxis=true, yaxis=true, xrange=[-2,2], yrange=[-.2,2],
       color=blue, parametric(t,abs(t),t,-2,2), label(["x=t, y=abs(t)",-.7,1.3]),
       color=red, nticks=100, parametric(cos(t),sin(t),t,0,%pi),
       label(["x=cos(t), y=sin(t)",0,1.1]),
       color=black, label(["Parametric",-1,1.75]))$
(%i4)  draw2d(xaxis=true, yaxis=true, xrange=[-2,2], yrange=[-.2,2],
       color=blue, implicit(y^2-x^2,x,-2,2,y,0,2), label(["y^2-x^2=0, y>=0",-.65,1.3]),
       color=red, implicit(x^2+y^2-1,x,-1,1,y,0,1), label(["x^2+y^2=1, y>=0",0,1.1]),
       color=black, label(["Implicit",-1,1.75]))$
```



4 Průběh funkce

Důležitou součástí vyšetřování průběhu funkce je určení intervalů, na kterých je tato funkce monotónní.

Vyšetřit průběh funkce f znamená určit:

- Definiční obor $D(f)$, body a intervaly spojitosti a nespojitosti.
- Sudost, lichost, periodicitu, resp. jiné speciální vlastnosti.
- Jednostranné limity v bodech nespojitosti, v hraničních bodech a v bodech $\pm\infty$.
- Nulové body; intervaly, na kterých je f kladná a záporná.
- f', stacionární body, lokální a globální extrémy; intervaly, na kterých je f rostoucí, klesající a konstantní.
- f'', inflexní body; intervaly, na kterých je f konvexní a konkávní.
- Asymptoty bez směrnice a asymptoty se směrnicí.
- Obor hodnot $H(f)$ a nastínit graf funkce.

Nejnázornější představu o průběhu funkce nám většinou poskytne graf. Při jeho konstrukci využíváme všechny zjištěné údaje. Mnohokrát jsou ale nedostatečné, proto je musíme doplnit vhodně zvolenými funkčními hodnotami.

Průběh funkce $f(x) = \frac{8(x-2)}{x^2} = \frac{8x-16}{x^2}$.

```
(%i1)  f(x):=(8*x-16)/x^2;
(%o1)  f(x) := (8x-16)/x^2
```

- $D(f) = R - \{0\} = (-\infty; 0) \cup (0; \infty)$.

Pomocí příkazu `denom` (denominator) zjistíme, kdy je jmenovatel nulový.

```
(%i3)  fm:denom(f(x));solve(fm=0,x);
(fmen) x^2
(%o3)  [x = 0]
```

- f není periodická, f není sudá, f není lichá.
- f je spojitá na intervalech $(-\infty; 0)$, $(0; \infty)$, v bodě 0 je nespojitá.
- $\lim\limits_{x\to\pm\infty} f(x) = \lim\limits_{x\to\pm\infty} \frac{8x-16}{x^2} = \lim\limits_{x\to\pm\infty} \left(\frac{8}{x} - \frac{16}{x^2}\right) = \frac{8}{\pm\infty} - \frac{16}{\infty} = 0 - 0 = 0.$

```
(%i5)  limit(f(x),x,minf);limit(f(x),x,inf);
(%o4)  0
(%o5)  0
```

- $\lim\limits_{x\to 0^-} f(x) = \lim\limits_{x\to 0^-} \frac{8(x-2)}{x^2} = \frac{-16}{0^+} = -\infty$, $\lim\limits_{x\to 0^+} f(x) = \lim\limits_{x\to 0^+} \frac{8(x-2)}{x^2} = \frac{-16}{0^+} = -\infty$.

```
(%i7)  limit(f(x),x,0,minus);limit(f(x),x,0,plus);
(%o6)  -inf
(%o7)  -inf
```

- Bod $x = 0$ je neodstranitelný bod nespojitosti II. Druhu.
- $x = 0$ je asymptota bez směrnice.
- $f(x) = \frac{8x-16}{x^2} = 0. \Leftrightarrow 8x - 16 = 0. \Leftrightarrow x = 2.$

Pomocí příkazu `num` (numerator) zjistíme, kdy je čitatel nulový.

```
(%i9)  fcit:num(f(x));solve(fcit=0,x);
(fcit) 8x − 16
(%o9)  [x = 2]
```

- $x = 2$ je nulový bod f. $\Rightarrow$ $\begin{cases} f(x) < 0 \text{ pro } x \in (-\infty; 0), \\ f(x) < 0 \text{ pro } x \in (0; 2), \\ f(x) > 0 \text{ pro } x \in (2; \infty). \end{cases}$

$f(2) = 0$, f není v bodě $x = 0$ definována.

$\Rightarrow$ Funkce f nemění znaménko na intervalech $(-\infty; 0)$, $(0; 2)$, $(2; \infty)$.

$\Rightarrow$ Stačí zvolit libovolný bod v daných intervalech a ověřit jeho hodnotu.

```
(%i13) f(2);f(-1);f(1);f(3);
(%o10) 0
(%o11) −24
(%o12) −8
(%o13) 8/9
```

- $f'(x) = \left[\frac{8x-16}{x^2}\right]' = \frac{8x^2-(8x-16)2x}{x^4} = \frac{32x-8x^2}{x^4} = \frac{32-8x}{x^3}$, $x \in R$, $x \neq 0$.

```
(%i15) f1(x):=diff(f(x),x,1)$ ratsimp(f1(x));
(%o15) −(8x−32)/x^3
```

- $f'(x) = \frac{32-8x}{x^3} = 0. \Leftrightarrow 32 - 8x = 0. \Leftrightarrow x = 4.$

```
(%i16) solve(f1(x)=0,x);
(%o16) [x = 4]
```

- f' je v bodě 0 nespojitá.

```
(%i18) f1men:denom(ratsimp(f1(x)));solve(f1men=0,x);
(f1men) x^3
(%o18) [x = 0]
```

- $x = 4$ je nulový bod f'. $\Rightarrow \begin{cases} f'(x) < 0,\ f \text{ je klesající pro } x \in (-\infty; 0), \\ f'(x) > 0,\ f \text{ je rostoucí pro } x \in (0; 4), \\ f'(x) < 0,\ f \text{ je klesající pro } x \in (4; \infty). \end{cases}$

$f'(4) = 0$, f' není v bodě $x = 0$ definována.

$\Rightarrow$ Funkce f' nemění znaménko na intervalech $(-\infty; 0)$, $(0; 4)$, $(4; \infty)$.

$\Rightarrow$ Stačí zvolit libovolný bod v daných intervalech a ověřit jeho hodnotu.

```
(%i22) subst(4,x,f1(x));subst(-1,x,f1(x));subst(1,x,f1(x));subst(5,x,f1(x));
(%o19) 0
(%o20) -40
(%o21) 24
(%o22) -8/125
```

- f má v bodě $x = 4$ lokální maximum i globální maximum $f(4) = 1$.

```
(%i23) f(4);
(%o23) 1
```

- f nemá lokální a ani globální minimum.
- $f''(x) = \left[\frac{32-8x}{x^3}\right]' = \frac{-8x^3-(32-8x)3x^2}{x^6} = \frac{16x^3-96x^2}{x^6} = \frac{16x-96}{x^4}$, $x \in R$, $x \neq 0$.

```
(%i25) f2(x):=diff(f(x),x,2)$ ratsimp(f2(x));
(%o25) (16*x-96)/x^4
```

- $f''(x) = \frac{16x-96}{x^4} = 0. \Leftrightarrow 16x - 96 = 0. \Leftrightarrow x = 6.$

```
(%i26)  solve(f2(x)=0,x);
(%o26)  [x = 6]
```

- f'' je v bodě 0 nespojitá.

```
(%i28)  f2men:denom(ratsimp(f2(x)));solve(f2men=0,x);
(f2men)  x^4
(%o28)  [x = 0]
```

- $x = 6$ je nulový bod f''. $\Rightarrow$ $\begin{cases} f''(x) < 0,\ f \text{ je konkávní pro } x \in (-\infty; 0), \\ f''(x) < 0,\ f \text{ je konkávní pro } x \in (0; 6), \\ f''(x) > 0,\ f \text{ je konvexní pro } x \in (6; \infty). \end{cases}$

$f'(6) = 0$, f'' není v bodě $x = 0$ definována.

$\Rightarrow$ Funkce f'' nemění znaménko na intervalech $(-\infty; 0)$, $(0; 6)$, $(6; \infty)$.

$\Rightarrow$ Stačí zvolit libovolný bod v daných intervalech a ověřit jeho hodnotu.

```
(%i32)  subst(6,x,f2(x));subst(-1,x,f2(x));subst(1,x,f2(x));subst(7,x,f2(x));
(%o29)  0
(%o30)  −112
(%o31)  −80
(%o32)  16/2401
```

- Bod $x = 6$ je inflexní bod funkce f.

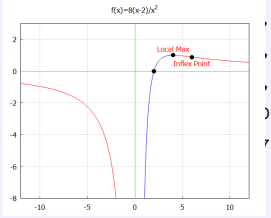
```
(%i33)  f(6);
(%o33)  8/9
```

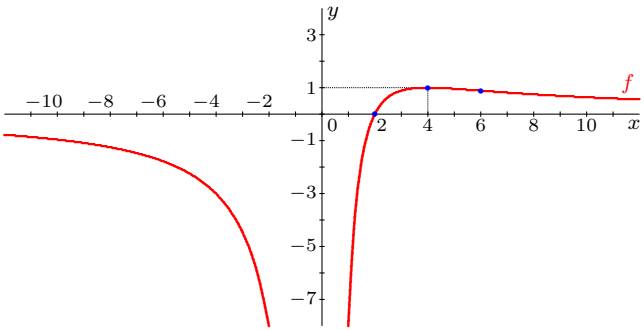
$\left.\begin{array}{l} \bullet\ k = \lim\limits_{x \to \pm\infty} \frac{f(x)}{x} = \lim\limits_{x \to \pm\infty} \frac{8x-16}{x^3} = \lim\limits_{x \to \pm\infty} \left(\frac{8}{x^2} - \frac{16}{x^3}\right) = 0 - 0 = 0. \\ \bullet\ q = \lim\limits_{x \to \pm\infty} [f(x) - kx] = \lim\limits_{x \to \pm\infty} [f(x) - 0 \cdot x] = \lim\limits_{x \to \pm\infty} f(x) = 0. \end{array}\right\} \Rightarrow y = kx + q = 0.$

```
(%i35) km:limit(f(x)/x,x,minf);kp:limit(f(x)/x,x,inf);
(km)   0
(kp)   0
(%i37) qm:limit(f(x)-km*x,x,minf);qp:limit(f(x)-kp*x,x,inf);
(km)   0
(kp)   0
```

- $y = 0$ je asymptota se směrnicí.
- $H(f) = (-\infty; 1\rangle$.

```
(%i38) draw2d(grid=true,xaxis=true,yaxis=true,xrange=[-12,12],yrange=[-8,3],
                                                  ,color=blue,explicit(f(x),x,0,4),
                                                  ,x,-12,0),explicit(f(x),x,4,12),
                                                  ,f(6)-.4],["Local Max",4,f(4)+.4]),
                                                  ),t,t,-8,3),parametric(t,0,t,-12,12),
                                                  ',points([[4,f(4)],[6,f(6)],[2,f(2)]]))$
```





Obr. 1: *Graf funkce* $f(x) = \frac{8(x-2)}{x^2}$

Literatura

1. Blaško R., *Matematická analýza I,* Žilina, EDIS 2009.
2. Blaško R., *Matematická analýza I,* skriptum, `http://frcatel.fri.utc.sk/~beerb/ma1/sa1.pdf`.

3. Blaško R., *Neurčitý a určitý integrál reálnej funkcie,* skriptum, http://frcatel.fri.utc.sk/~beerb/ma1/sa2.pdf.

4. Blaško R., *Základy lineárnej algebry a základy matematickej analýzy pre manažérov,* skriptum, http://frcatel.fri.utc.sk/~beerb/ma1/zla-zma.pdf.

5. Buša J., *Maxima Open source systém počítačovej algebry,* online, http://people.tuke.sk/jan.busa/kega/maxima/maxima.pdf, 2006.

6. Bittinger M. L., Ellenbogen D. J., Surgent S. A., *Calculus and its Applications,* Addisok-Wesley, ISBN-10: 0-321-69433-3.

7. Crowell B., *Calculus, Light and Matter,* www.lightandmatter.com, March 2010.

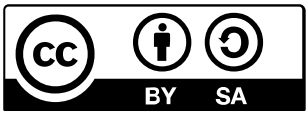
8. Hannan Z., *wxMaxima for Calculus I and II,* Solano Community College, https://wxmaximafor.wordpress.com/.

9. Mardsen J., Weinstein A., *Calculus I–III,* Springer.

10. Strang G., Calculus, *Wellesley-Cambridge Press,* Box 82-279 Wellesley MA 02181.

Autor

Rudolf Blaško, Katedra matematických metód a operačnej analýzy, Fakulta riadenia a informatiky, Žilinská univerzita v Žiline, Vysokoškolákov 8215/1, 010 26 Žilina, Slovenská republika, e-mail: beerb@frcatel.fri.uniza.sk



Wireless Signal Processing in GNU Radio Environment

Remigiusz Olejnik

1 Introduction

This chapter presents a GNU Radio — an open source software package aimed to be used as an environment for wireless signal processing. Such application could be useful for computer science students during their education process in the courses related with data transmission, telecommunications, radio communications etc.

2 What is GNU Radio?

GNU Radio [2]:

1. is a free and **open-source** software development toolkit that provides signal processing blocks to implement software radios;
2. can be used with readily-available low-cost external RF hardware (such as RTL-SDR or HackRF) to create software-defined radios;
3. or even without hardware — in a simulation-like environment;
4. is widely used in hobbyist, academic and commercial environments to support both wireless communications research and real-world radio systems;
5. is a framework that enables users to design, simulate, and deploy highly capable real-world radio systems;
6. is a highly modular, "flowgraph"-oriented framework that comes with a comprehensive library of processing blocks that can be readily combined to make complex signal processing applications;
7. has been used for a huge array of real-world radio applications, including mobile communications, tracking satellites, radar systems, GSM networks, Digital Radio Mondiale, and much more — all in computer software;
8. can be used to develop implementations of basically any (band-limited) communication standard.

DOI: https://doi.org/10.11118/978-80-7509-898-6-0077

3 Why would I want GNU Radio?

Formerly the engineer had to develop a specific circuits for detection, decoding, encoding of a specific signal and debug all of them using costly equipment. Software-Defined Radio (SDR) approach moves the analog radio signal processing — as far as physically and economically feasible — to a computer, using algorithms in software. The engineer had to self-implement all DSP algorithms while in GNU Radio environment a user is capable of using highly optimized and peer-reviewed scalable implementations along with GUIs.

GNU Radio:

1. is a framework dedicated to writing signal processing applications for general purpose computers;
2. wraps functionality in easy-to-use reusable blocks;
3. offers excellent scalability;
4. provides an extensive library of standard algorithms;
5. is heavily optimized for a large variety of common platforms;
6. comes with a large set of examples to get you started from.

4 A flowgraph-based approach to Digital Signal Processing

In GNU Radio framework individual processing stages such as filtering, correction, analysis, detection etc. are represented by processing blocks; these blokcs are connected using simple flow-indicating arrows — see example in Fig. 4

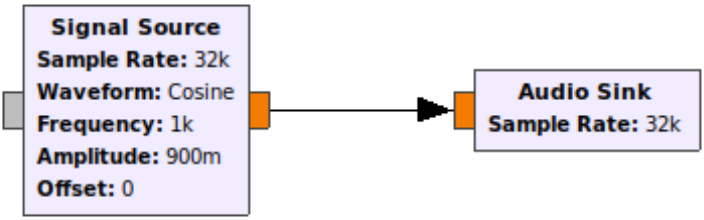


Fig. 1: *GNU Radio — two blocks (**Signal Source** and **Audio Sink**) connected with an arrow showing flow of the signal data*

A digital signal processing application is a complete graph of blocks, in GNU Radio called as **flowgraph**. Fig. 4 shows an example of flowgraph.

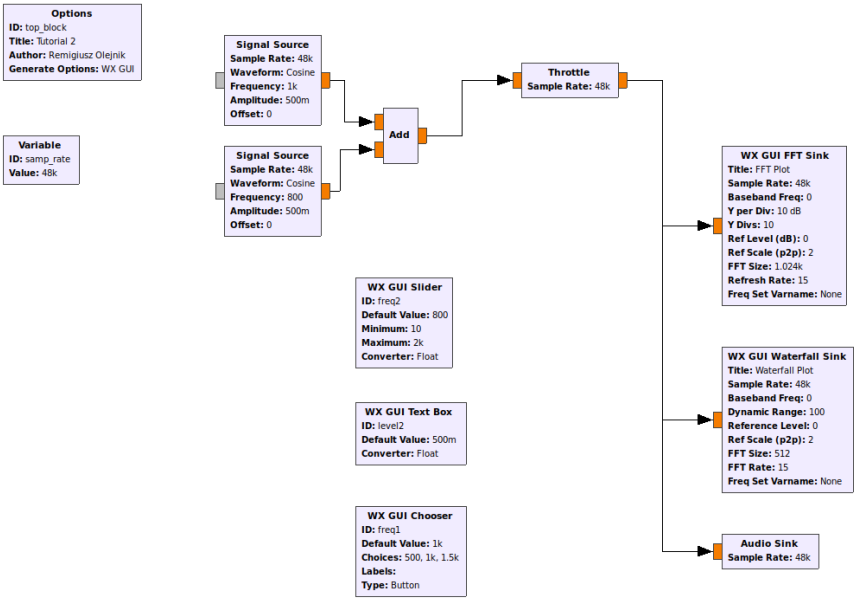


Fig. 2: *GNU Radio — seven blocks connected together form a flowgraph*

GNU Radio framework allows developing these processing blocks and creating flowgraphs, which comprise radio processing applications. Existing blocks could be combined into a high-level flowgraph.

5 Most popular GNU Radio blocks

GNU Radio comes with a large set of existing blocks. Most popular ones are presented below.

- Waveform Generators
  - Constant Source
  - Noise Source
  - Signal Source (e.g. Sine, Square, Saw Tooth)
- Modulators
  - AM Demod
  - Continuous Phase Modulation

  - PSK Mod / Demod
  - GFSK Mod / Demod
  - GMSK Mod / Demod
  - QAM Mod / Demod
  - WBFM Receive
  - NBFM Receive
- Instrumentation
  - Constellation Sink
  - Frequency Sink
  - Histogram Sink
  - Number Sink
  - Time Raster Sink
  - Time Sink
  - Waterfall Sink
- Math Operators
  - Abs
  - Add
  - Complex Conjugate
  - Divide
  - Integrate
  - Log10
  - Multiply
  - RMS
  - Subtract
- Channel Models
  - Channel Model
  - Fading Model
  - Dynamic Channel Model
  - Frequency Selective Fading Model
- Filters
  - Band Pass / Reject Filter
  - Low / High Pass Filter
  - IIR Filter
  - Generic Filterbank
  - Hilbert
  - Decimating FIR Filter
  - Root Raised Cosine Filter
  - FFT Filter
- Fourier Analysis

- FFT
- Log Power FFT
- Goertzel (Resamplers)
- Fractional Resampler
- Polyphase Arbitrary Resampler
- Rational Resampler (Synchronizers)
- Clock Recovery MM
- Correlate and Sync
- Costas Loop
- FLL Band-Edge
- PLL Freq Det
- PN Correlator
- Polyphase Clock Sync

6 RTL-SDR based WFM receiver example

In Fig. 6 a simple example of the broadcast WFM receiver is resented. It consists of **RTL-SDR Source** block as a radio signal source, **FM Demod** block as a FM demodulator, **Multiply Const** block supplying a volume value for the audio level and **Audio Sink** block that allows playing audio signal.

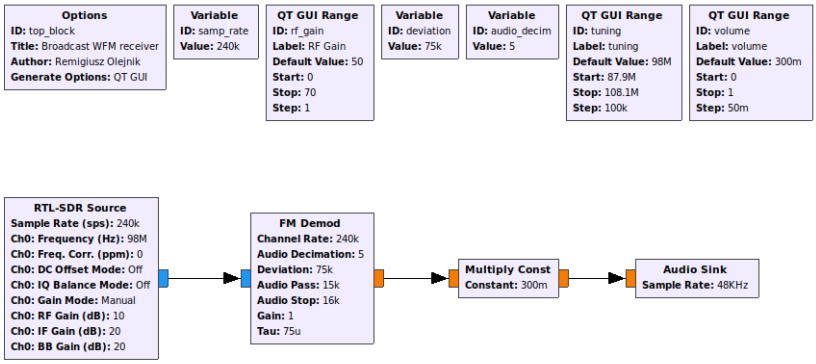


Fig. 3: *An example of the broadcast WFM receiver in GNU Radio*

7 Summary

GNU Radio is a free and open source software development toolkit that provides signal processing blocks to implement Software Defined Radios (SDRs). It is a highly mod-

ular, „flowgraph”-oriented framework, that comes with a large set of existing blocks. GNU Radio can be used with readily-available low-cost external RF hardware (such as RTL-SDR or HackRF) to create software-defined radios.

References

1. http://ioscs.zut.edu.pl/
2. GNU Radio, https://www.gnuradio.org/

Author

dr. inż. Remigiusz Olejnik, Zachodniopomorski Uniwersytet Technologiczny w Szczecinie.

Mobile Application Development

Radosław Maciaszczyk

The market of mobile devices especially smartphones is growing very fast. Numerous users of these devices causes that the mobile application market is also growing very fast. Universities and colleges have noticed this trend and courses related to mobile applications programming are present in most computer science curricula.

On the market we have a choice between practically two systems, Android and iOS. In Q3 2021, Android is installed on over 72% of devices [1]. Due to greater popularity and lower cost of developing applications, this course concerns programming in Android environment.

Android is an open source operating system for mobile devices and a corresponding open source project led by Google [2]. Android Open Source Project (AOSP) repository offer the information and source code needed to create custom variants of the Android OS, port devices and accessories to the Android platform, and ensure devices meet the compatibility requirements that keep the Android ecosystem a healthy and stable environment for millions of users.

As an open source project, Android's goal is to avoid any central point of failure in which one industry player can restrict or control the innovations of any other player. To that end, Android is a full, production-quality operating system for consumer products, complete with customizable source code that can be ported to nearly any device and public documentation that is available to everyone.

There are not much requirements to start developing applications. Generally you need knowledge of object oriented language, For Android it is (JAVA, KOTLIN).

The freeware Android Studio is used for programming. Development environment is available for multiple platforms including Windows, Mac, Linux, Chrome OS [3].

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA . On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as [4]:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices

DOI: https://doi.org/10.11118/978-80-7509-898-6-0083

- Apply Changes to push code and resource changes to your running app without restarting your app
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

Android is an operating system that is constantly evolving. This is undoubtedly an advantage, but continuous improvement requires constant development of programming skills. However, it is possible to distinguish the basic range of knowledge needed to develop applications. The following elements are isolated in the course provided by this project:

- System components
- User interaction
- Sensor handling
- Data exchange

The course consists of 45 hours of classes taught in lecture (15h), laboratory (15h) and project (15h).

Lecture:

- Introducing to mobile device and mobile systems.
- Application Fundamentals, components – activities, services, broadcast receivers, content providers
- Component lifecycle – activity, fragments, services
- User Interface, Introduction to Material Design, typography, main component
- Sensors, GNNS – use case of sensors, type of sensors, sensors lifecycle
- Threads and Services, Class – Runnable, Jobs Scheduler, Intent Service, AsyncTask
- Data persistence – Room Database, Using SD Card, SharedPreference class
- Networking, using sockets and HTTP connections
- Google firebase for Android – Cloud Messaging, Cloud firestore,
- Android Performance

Laboratory:

- Configure the Development Environment, Create first program. Debug programs
- Create interactive user interface. Introduction to widgets
- Activities and Intents
- Layouts. Using RecyclerView to display data
- Data persistence
- Sensors and Location
- Services, Notifications
- Networking

Project:

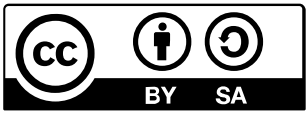
- Introduction to project, project's functions
- Working with own Project
- Documentation
- Presentation project

References

1. https://gs.statcounter.com/os-market-share/mobile/worldwide
2. https://source.android.com/
3. https://developer.android.com/studio
4. https://developer.android.com/studio/intro

Author

dr inż. Radosław Maciaszczyk, West Pomeranian University of Technology in Szczecin, e-mail: Radoslaw.Maciaszczyk@zut.edu.pl



Výukový kurs programovacího jazyka Lua

Tutorial for Programming Language Lua

Tomáš Hála

Abstrakt: Příspěvek představuje důvody vzniku nového předmětu o programovacím jazyce Lua a seznamuje s prvními poznatky z pilotního kursu vybraných částí. Návrh sylabu tohoto předmětu je rovněž připojen.

Tvorba nového kursu je řešena jako součást projektu *Innovative Open Source Courses for Computer Science*.

Klíčová slova: programovací jazyk Lua, kurs programování, open source software, výuka na vysoké škole

Abstract: This paper presents the reasons for the creation of a new course on programming language Lua and introduces the first findings from the pilot course of selected parts. A draft syllabus for this course is also attached.

The creation of the new course is being addressed as part of the *Innovative Open Source Courses for Computer Science*.

Key words: programming language Lua, programming course, open source software, university education

1 Úvod

Programování patří mezi stálé a neodmyslitelné součásti přípravy studentů informatických oborů. Volba programovacích jazyků, které jsou využívány ve výuce algoritmizace a programování na vysokých školách, vychází z historických tradic, z možností školy i z profilů absolventů a jejich předpokládaného pracovního zařazení, avšak souhrnně lze říci, že se většinou jedná o jazyky vhodné spíše pro produkční programování než o jazyky vhodné k výuce, a to i za cenu, že jsou zatíženy mnoha úskalími (syntaktický zápis, množství knihoven), které komplikují přímočarou cestu k pochopení algoritmu.

Upřednostňování „okamžitého“ výstupu v podobě znalosti produkčního jazyka může být jedním z důvodů, proč méně známé jazky jsou zastoupeny ve výuce programování jen zřídka. V nabídce předmětů vysokých škol lze nalézt pak jen předměty zabývající se konkrétním programovacím jazykem, avšak programovací jazyk Lua není dosud v České republice vyučován ani touto formou, což je jedním z důvodů, proč byl kurs progra-

DOI: https://doi.org/10.11118/978-80-7509-898-6-0086

mování v jazyce Lua zařazen do projektu *Innovative Open Source Courses for Computer Science*.

Potřebu samostatného předmětu pro výuku programovacího jazyka Lua násobí i fakt, že na Provozně ekonomické fakultě Mendelovy univerzity v Brně vzniklo v minulých letech několik bakalářských a diplomových prací, které využily programovací jazyk Lua pro řešení zadaného tématu, například pro syntaktickou kontrolu textů v ConTEXtu (Hanuš, 2015; Makeš, 2015) nebo pro výpočty ve vexilografickém modulu (Treťák; 2016) či dvojice Kocurová (2020) a Kašparová (2020), které využily programovací jazyk Lua pro výpočty a řízení vykreslování statistických grafů. Všichni studenti byli nuceni se jazyk naučit sami, neorganizovaně a jen s podporou vedoucího práce, což by nebylo po zavedení takového předmětu nutné a přípravu studentů pro práci s jazykem Lua v rámci závěrečných prací výrazně zkvalitnilo.

2 Programovací jazyk Lua

Programovací jazyk Lua, ač méně známý, nepatří mezi tzv. „výkřiky poslední módy“ – jedná se o programovací jazyk, jenž vznikl v 90. letech minulého století. Jeho první verze byla vytvořena již v roce 1993.

Byl navržen jako jazyk skriptovací a řadíme jej do kategorie vysokoúrovňových jazyků, multiparadigmatických, imperativní a procedurální programovací jazyků. Je charakterizován jako jazyk odlehčený. Jeho významnou vlastností je též reflexivita.

Srovnáme-li programovací jazyk Lua s ostatními programovacímu jazyky, zjistíme, že stojí někde „na půl cesty“ mezi jazyky vhodnými pro výuku (např. Pascal) a jazyk produkčními (např. C, C++, Java). Syntakticky se podobá v řadě rysů srozumitelnému a původně pouze pro účely výuky konstruovanému programovacímu jazyku Pascal (např. Wirth, 1981), což umožňuje velmi rychle osvojení syntaktických konstrukcí, na druhou stranu neobsahuje některé užitečné rysy, například striktní typové kontroly, které velmi účinně brání uživateli zapisovat nesmyslné výrazy.

Již zmíněné snadné osvojení jazyka je – kromě souborů – podpořeno i jediným *typickým* strukturovaným datovým typem `tabulka` (`table`), sloužící jako indexované i asociativní pole, a to jak homogenní, tak heterogenní. Pomocí něho lze pak simulovat i další strukturované datové typy – množinu, multimnožinu, lineární seznam, stromové struktury atd.

Možností využití programovacího jazyka ve výuce se zabývala snad jen jediná práce (Vévoda, 2015), porovnávající možnosti dříve používaného jazyka Pascal a jazyka Lua. Po zhodnocení výhod a nevýhod využití jazyka Lua ve výuce nevylučuje, pokud se přijme

fakt absence striktní kontroly datových typů a dalších jevů souvisejících s odlehčením jazyka.

3 Obsah kursu

Tématické celky potřebné pro zvládnutí programovacího jazyka Lua se, jak je obvyklé, skládají z výkladové (levý sloupec) i z praktické části (pravý sloupec).

Programování v jazyce Lua

1. Úvod a popis jazyka Lua

   - historie
   - datové typy, hodnoty, výrazy
   - čísla, boolean, nil, řetězce
   - proměnné, přiřazení
   - konverze mezi datovými typy
   - vstup a výstup
   - matematické funkce

2. Operátory

   - aritmetické operátory
   - bitové operátory
   - logické operátory
   - relační operátory
   - spojení řetězců
   - vyhodnocování výrazů
   - priorita operátorů

3. Konstrukce programovacího jazyka Lua

   - podmínky a příkazy větvení
   - příkazy cyklu s podmínkou – příkazy while, repeat/until
   - příkaz cyklu for
   - zpracování dat

4. Řetězce

   - operace s řetězci
   - kódování UTF-8
   - syntaktické zkratky
   - vzory pro hledání
   - captures

5. Tabulky

- pole, asociativní pole
- přístup k prvkům asociativního pole
- řazení, modifikace řazení
- vytvoření tabulky
- zjištění počtu prvků
- operace s tabulkou

6. Funkce

- deklarace, volání, návratové hodnoty
- parametry, volitelné parametry
- rekurze
- rekurzivní algoritmy
- serializace, výpis struktury

7. Funkce

- funkce jako datový typ
- funkce předávané jako parametry
- iterátory, uzávěry
- využití vnějších podprogramů
- konstrukce vlastního iterátoru

8. Textové soubory

- popis a vlastnosti textových souborů
- operace s textovými soubory
- použití různých režimů zpracování textových souborů

9. Binární soubory

- popis a vlastnosti binárních souborů
- operace s binárními soubory
- konverze binárních dat na textová a opačně
- přímý přístup k datům

10. Moduly

- struktura a použití modulu
- diagram signatury
- návrh uživatelského abstraktního datového typu
- implementace abstraktního datového typu

11. Komunikace s OS

- příkazový řádek
- systémové proměnné
- volání příkazů
- datum a čas
- knihovna os
- zpracování konfiguračních souborů

12. Využití jazyka lua v aplikacích

- principy využití
- popis aplikací
- Lua a programovací jazyk C
- Lua v ConTEXtu
- využití při tvorbě her

4 Použité technologie

4.1 Verze programovacího jazyka

Verze programovací jazyka Lua není rozhodující pro zvládnutí základů tohoto jazyka. Lze však doporučit verze 5.3 a vyšší.

4.2 Vývojové prostředí

Před započetím výuky je potřeba rozhodnout o tom, které vývojové prostředí bude doporučeno, případně vyžadováno.

Volba vývojového prostředí je velmi často závislá na preferencích uživatele-programátora, ale i na operačním systému a možnostech instalace vývojových prostředí. Proto nelze dát jednoznačnou odpověď na to, který způsob práce je nejvhodnější. Omezíme se zde tedy jen na přehled možností a získané poznatky.

Jako zastánce využití volně dostupného programového vybavení v co nejširší míře budu jako první volbu vždy zkoumat, jaké možnosti použití nabízí **operační systém Linux**. Tento přístup naplňuje poslání projektu *Innovative Open Source Courses for Computer Science*, v němž sylabus kursu Programování v jazyce Lua vzniká, i v širším slova smyslu.

Pro použití pod OS Linux postačuje přítomnost libovolného textového editoru (joe, vim, emacs, nano, ...) a přítomnost interpretu jazyka Lua, obojí lze bez dalších komplikací spouštět z příkazového řádku, případně s možným využitím přesměrování standardního vstupu a výstupu.

Pro výuku lze též použít i některé **aplikace přístupné on-line** – ty však mají různé vlastnosti a ne vždy dané prostředí může vyhovovat dané situaci, ať již hovoříme opět o preferencích studentů, nebo o potřebách řešených úloh.

Zcela nekomfortním se projevil on-line překladač na stránkách `www.lua.org`. I přesto však bohatě postačuje pro demostraci jednoduchých ukázek, například pro vyhodnocování výrazů. Nesporně jej lze doporučit také pro ověření funkčnosti kódu, neboť se

jedná o aplikaci z referenčních stránek programovacího jazyka Lua, na jejichž obsahu se podílejí autoři jazyka.

Pro operační systém **Windows** existuje několik integrovaných vývojových prostředí, které umožňují práci s řadu programovacích jazyků Lua. Ten však nebývá zařazen mezi výchozí instalované jazyky, a proto je potřeba jej instalovat dodatečně. I zde je potřeba rozvážit, které z integrovaných vývojových prostředí splňuje potřeby řešených úloh a poskytuje patřičné pohodlí, a podle toho přijmout rozhodnutí co doporučit studentům.

5 Diskuse a závěr

V kursu Programování v jazyce Lua, který je připravován v rámci mezinárodního projektu *Innovative Open Source Courses for Computer Science*, jsou využity zkušenosti z dlouholeté výuky programování v různých programovacích jazycích uskutečňované v rámci studia informatických předmětů na Provozně ekonomické fakultě Mendelovy univerzity v Brně. Navržený sylabus obsahuje nosná témata, která vedou studenta k pochopení programovací jazyka Lua, demonstrovaná na adekvátních sadách příkladů.

Omezená časová dotace pro běh kursu v rámci Letní školy neumožňuje se studenty probrat všechna připravená témata, i přesto tento první běh pracovní verze kursu umožnil získat ohlasy od studentů. Ač byly vesměs příznivé, řada studentů poskytla zajímavé postřehy a podněty, které budou po kritickém zhodnocení následně reflektovány jednak úpravou sylabu, jednak doplněním a úpravou připravovaných studijních materiálů.

Jedním z takových doplňků, který bude zařazen do náplně cvičení, se stane „vlastnoručně“ sestavená funkce pro vizualizaci datových struktur a v nich uložených hodnot, což se při pilotním běhu kursu ukázalo jako nezbytné pro zvýšení názornosti výkladu.

Návrh sylabu kursu předpokládá budoucí zařazení mezi vysokoškolské předměty vyučované pro informatické obory na Provozně ekonomické fakultě Mendelovy univerzity v Brně. Jeho první běh by se měl uskutečnit v letním semestru akademického roku 2022/2023.

Po zavedení kursu lze očekávat následující přínosy:

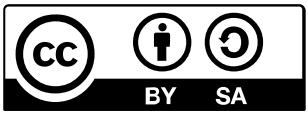
- využití nástroje, kterým lze efektivněji vysvětlit principy některých algoritmů;
- získání nástroje pro snadnou tvorbu jednoduchých programů pro zpracování dat, ať již pro vlastní potřebu, nebo pro více uživatelů;
- podporu pro studenty, kteří chtějí vyvíjet programové vybavení v jazyce Lua s využitím například v typografickém systému ConTEXt nebo pro vývoj her;
- zvýšení povědomí o oblasti open source a jejím využití v praxi.

Literatura

1. Ierusalimschy, R. *Programming in Lua.* Rio de Janeiro : Lua.org, 2016. 369 s. ISBN 978-85-903798-6-7.
2. Hanuš, A. *Syntaktický analyzátor zdrojových textů ve formátu ConTEXt.* Diplomová práce. Brno : Mendelova univerzita v Brně, 2015. 83 s.
3. Kašparová, A. *Implementation of module drawing charts for ConTEXt* Bakalářská práce. Brno : Mendelova univerzita v Brně, 2020.
4. Kocurová, T. *Implementation of module drawing charts for ConTEXt* Bakalářská práce. Brno : Mendelova univerzita v Brně, 2020.
5. *Lua 5.4 Reference Manual [on-line].* [Rio de Janeiro] : Lua.org, PUC-Rio, 2022. Dostupné na: `http://www.lua.org/manual/5.4/`.
6. Makeš, D. *Detekce a korekce typografických jevů ve značkovaném textu.* Diplomová práce. Brno : Mendelova univerzita v Brně, 2015. 81 s.
7. Treťák, M. *Tvorba vexilografického modulu s využitím jazyků MetaPost a Lua.* Bakalářská práce. Brno : Mendelova univerzita v Brně, 2016. 50 s.
8. Vévoda, P. *Programovací jazyk Lua a možnosti jeho využití v předmětu Programovací techniky.* Bakalářská práce. Brno : Mendelova univerzita v Brně, 2015. 70 s.
9. Wirth, N. *Algoritmy a štruktury údajov.* Bratislava : Alfa, 1988. 488 s.

Autor

RNDr. Tomáš Hála, Ph.D., Mendelova univerzita v Brně, Provozně ekonomická fakulta, ústav informatiky, Zemědělská 1, CZ 613 00 Brno, e-mail: thala@mendelu.cz



Nástroje open source pro zpracování textů

Open Source Tools for Text Processing

Jiří Rybička

Abstrakt: V rámci projektu Innovative Open Source Courses for Computer Science je řešen kurz zabývající se zpracováním textů pomocí počítače. Cílem příspěvku je představit hlavní motivy zvolené koncepce tohoto předmětu, stručně seznámit s nejdůležitějšími prvky a inovativním pohledem na problematiku a uvést ukázky vybraných částí návrhu předmětu.

Klíčová slova: open source, zpracování textů, koncepce VŠ výuky, technologie na bázi TEX

Abstract: It is being solved within the project Innovative Open Source Courses for Computer Science computer-based word processing course. The aim of the paper is to introduce the main motives of the chosen concept of this subject, briefly acquaint with the most important elements and an innovative view of the issue and state examples of selected parts of the course design.

Key words: open source, text processing, college teaching concept, technology based on TEX

1 Úvod

Zpracování textů pomocí programového vybavení osobních počítačů má dlouholetou tradici a vyznačuje se masivním zastoupením ve všech oblastech, kde se mohou počítače využívat. Dlouhodobě se programy pro zpracování textů umisťují mezi nejpoužívanějšími aplikacemi, v posledních letech je téměř samozřejmostí, že v každém počítači je instalován některý z kancelářských balíků, jehož součástí a nejpoužívanější aplikací je textový procesor.

Na tuto situaci již také zareagovala norma ČSN 01 6910 s názvem „Úprava dokumentů zpracovaných textovými procesory“ (ČSN 01 6910, 2014). Není zde přímo uvedeno, o jaký procesor se jedná, z kontextu a odkazů na některé funkce však můžeme usoudit, že jde o textový procesor Word firmy Microsoft, případně o podobně se chovající textové procesory z kancelářských balíků z oblasti open source – Open Office Writer, Libre Office Writer. Kromě toho lze v některých případech uvažovat i cloudová řešení (Google, Microsoft), která jsou používána stále častěji a představují značný benefit především při kooperaci na dokumentech a jejich sdílení.

DOI: https://doi.org/10.11118/978-80-7509-898-6-0093

Základní práce s textovým procesorem nepochybně patří do výuky, nikoliv však na vysokoškolské úrovni. V běžně praktikovaných rámcových vzdělávacích programech je tato úroveň zahrnuta do základního školství a je určitým způsobem přítomna i na školách středních. Vzhledem k masivnímu rozšíření těchto textových procesorů je samozřejmé, že tomu tak bude i po poměrně zásadní změně koncepce výuky informatiky na základních a středních školách. Můžeme si v této chvíli přirozeně položit zásadní otázku, proč a jak zpracování textů realizovat jako vysokoškolský předmět.

Problematika zpracování textů má několik rovin:

1. Zmíněný obsah vyučovaný na nižších stupních škol lze zařadit do roviny *základního ovládání* určitého programového systému – poznání nabídek, některých funkcí, editačních příkazů textu atd.
2. Druhou rovinou je účel a smysl jednotlivých nástrojů – *typografická a jazyková pravidla* vyúsťující v souhrnné požadavky na cílový tvar dokumentu.
3. Konečně třetí rovinou je *technická efektivnost* zpracování dokumentů zahrnující zejména rozšíření možností snadných úprav a znovupoužití zpracovaného materiálu.

2 Koncepce výuky zpracování textů

Ve vysokoškolské výuce se předpokládá, že obsah předmětu navazuje na určité předchozí znalosti a dovednosti, ale zároveň se rozvíjejí další, vyšší a komplikovanější prvky vyžadující rozšířený, zobecněný a celistvý pohled na problematiku. Proto se logicky do koncepce předmětu z oblasti zpracování textů na vysokoškolské úrovni promítá především část roviny druhé (část neobsažená v předchozích vzdělávacích stupních) a téměř celá rovina třetí.

Zároveň je stěžejní součástí také celkový a zobecněný pohled na metodiku zpracování – oddělení jednotlivých fází a použitých (použitelných) nástrojů. Specificky ve vysokoškolské variantě je v této oblasti kladen důraz na odborné a vědecké dokumenty, protože se s nimi studenti setkávají, potřebují je vytvářet a potřebují k tomu dostatek informací.

Významným prvkem konkrétně s ohledem na projekt inovace studijních předmětů za použití nástrojů z oblasti open source je dosazení volně dostupných komponent do celého procesu zpracování textů. Zde se jedná nejčastěji o použitelná písma a vhodné programové vybavení. Za přínosný lze považovat i záměr vybavit studenty informacemi o různých variantách tohoto vybavení, aby byli schopni aplikovat na určitý druh dokumentu optimální způsob jeho zpracování, tj. vybrat takový software, který co nejefektivnějším způsobem podpoří realizaci výsledného díla.

3 Obsah kurzu a nejdůležitější součásti

Z uvedených skutečností vychází obsah konstruovaného kurzu. Jako softwarová podpora je nejvíce diskutován systém postavený na principu TEXu. Hlavním důvodem pro tuto volbu je orientace na odborné a vědecké dokumenty, pro něž je tento systém ve většině případů dostatečně efektivní a umožňuje realizovat i velmi náročné typografické a technické požadavky. Protože se nejedná o masově rozšířený textový procesor typu Writer nebo Word, nepředpokládá se, že by se s ním studenti v předchozí výuce setkali. Nutně se zde proto musí věnovat určitý prostor i první zmíněné rovině, tj. základnímu ovládání komponent, které k systému patří.

3.1 Seznam témat kurzu

Návrh sylabu kurzu počítá s 12 kapitolami, v každé z nich je prostor pro teoretickou část, na niž pak navazuje praktická část – cvičení u počítače. Počet kapitol odpovídá zhruba počtu využitelných týdnů typického jednosemestrálního vysokoškolského kurzu. Přibližný obsah je uveden v následujícím přehledu:

1. **Dokument a metoda jeho zpracování**
   Teoretická témata: • Dokumentní prvky – princip • Identifikace prvků v dokumentu • Typografický návrh dokumentu – reprezentace prvků • Technologie – realizace typografického návrhu • Technologický princip na bázi TEXu • Technologický princip na bázi systémů open office
   Praktické cvičení: • Systém TEX – základní principy • Distribuce, instalace • Editory, první dokument, překlad, protokol o překladu
2. **Základní parametry dokumentu**
   Teoretická témata: • Knižní písmo, volba typu písma • Základní písmo, stupeň základního písma • Elektronický/tištěný dokument, formát (rozměry) stránek • Technologie – definice makropříkazů
   Praktické cvičení: • Zdroje písem, přehled dostupných písem, příklady • Parametry základního písma, volba základního písma • Definice maker s parametry, přístup LATEX, přístup TEX
3. **Speciální znaky, národní prostředí**
   Teoretická témata: • Kódování dokumentu • Nastavení národního prostředí (jazykově závislé texty, dělení slov) • Nastavení dělicího algoritmu • Speciální znaky a jejich řešení • Technologie – délkové jednotky, měrné typografické systémy
   Praktické cvičení: • Kódování UTF-8, speciální znaky, vkládání kódů • Vzory dělení slov • Práce s délkovými jednotkami, výpočet délek, měření délek
4. **Sazba odstavců, algoritmy, parametry**

Teoretická témata: ● Základní text – parametry odstavců (zarážky × odsazení; zarovnání) ● Odstavcové prvky jiné než základní text – parametry (citáty, výčty) ● Technologie – délky, délkové registry, operace s délkami
Praktické cvičení: ● Parametry odstavců, hladká sazba s různými parametry, příklady ● Zvláštní odstavce – odrážkové a číslované seznamy, citáty ● Délkové registry, aditivní a multiplikativní operace s registry

5. **Smíšená sazba**
   Teoretická témata: ● Vyznačování ● Využití doplňkového typu písma ● Využití různých řezů písma (kromě vyznačovacích) ● Barva písma a její využití (technologie – barvy, modely, definice)
   Praktické cvičení: ● Písmové řezy ● Různé typy písma v jednom dokumentu, volba kompatibilních fontů ● Práce s barvami (definice uživatelských barev, barevné modely)
6. **Členění dokumentu**
   Teoretická témata: ● Systémy mezinadpisů ● Iniciály ● Tvorba obsahu ● Technologie – číslování (čítače, reference)
   Praktické cvičení: ● Předdefinované mezinadpisy, uživatelské definice ● Technologie iniciál ● Čítače a křížové odkazy na ně
7. **Stránky**
   Teoretická témata: ● Odstavec a stránkový zlom ● Stránková záhlaví a paty ● Poznámky pod čarou ● Marginálie ● Stránkový design spec. stránek (titul, vyd. záznam, tiráž)
   Praktické cvičení: ● Odstavcové parametry pro optimální stránkový zlom ● Vkládání poznámek pod čarou, vkládání marginálií ● Úprava stránky s různými řezy a velikostmi písma
8. **Matematické a podobné výrazy**
   Teoretická témata: ● Prvky výrazů ● Textová a vysazená matematika ● Začlenění výrazů do dokumentu, křížové odkazy
   Praktické cvičení: ● Přehled matematických prvků (exponenty, indexy, zlomky...) ● Prostředí pro matematickou sazbu a jejich možnosti ● Výrazy se sumami, limitami a maticemi
9. **Tabulky**
   Teoretická témata: ● Typy tabulek ● Způsoby zarovnání tab. obsahu ● Začlenění tabulky do dokumentu – plovoucí/neplovoucí objekty, popisky
   Praktické cvičení: ● Prostředí `tabbing` a `tabular` ● Zarovnání číselných dat v tabulkách ● Cvičení s různými typy tabulek
10. **Obrazový materiál a grafika**

Teoretická témata: ● Typy obrazů – podle bar. hloubky, podle zdroje ● Grafické prvky v dokumentu ● Technologie – možnosti pořizování grafických prvků nástroji systému ● Požadované vlastnosti grafických prvků importovaných zvnějšku ● Popisky obrázků, vazba na popisky tabulek, plovoucí/neplovoucí objekty
Praktické cvičení: ● Příprava grafiky – rastrový formát, vektorový formát ● Možnosti vektorových formátů, vkládání souborů PDF ● Prostředí `picture` ● Prostředí pro vkládání tabulek a obrázků

11. **Dokument**
Teoretická témata: ● Sestava stránek ● Obsahy, rejstříky, křížové odkazy ● Vyřazení stránek pro tisk, vazby, zpracování tištěného dokumentu
Praktické cvičení: ● Návrh stránkových prvků: běžná záhlaví, marginálie, číslování stránek ● Technologie obsahu, seznamů tabulek a obrázků ● Uspořádání stránek, vyřazení stránek pro tisk více stránek na jeden list
12. **Návrh a realizace vlastního dokumentu**
Teoretická témata: ● Procvičení typografického návrhu a technické realizace celého dokumentu
Praktické cvičení: ● Typografický návrh ● Určení prvků dokumentu ● Technologické zpracování (styly, makra)

3.2 Komentář k první kapitole

Témata první kapitoly se věnují základnímu konceptu přístupu ke zpracování dokumentů, proto se tomuto obsahu budeme věnovat poněkud podrobněji. Východiskem zobecněné metody zpracování dokumentů je dlouhodobě funkční (až historická) dělba práce rozdělená mezi nejméně tři osoby (možná přesněji pracovní role):

1. **Autor** – dodává obsahový materiál dokumentu a určuje jeho prvky (vyznačení, nadpisy, poznámky, popisky atd.).
2. **Úpravce (designer)** – určuje vzhled dokumentu stanovením parametrů jednotlivých prvků definovaných autorem.
3. **Sazeč** – podle parametrů stanovených úpravcem realizuje dokument a jeho prvky v dané technologii.

To, že v dnešní době tyto role často splývají do jedné fyzické osoby, vůbec neznamená, že neexistují, spíše naopak – zanedbáváním některých aspektů těchto rolí přirozeně a zákonitě vznikají dokumenty s řadou zásadních nedostatků. Z rámcového popisu práce zmíněných osob (rolí) vyplývá také, jak na sebe činnosti logicky navazují. Je proto neefektivní, jsou-li tyto činnosti prováděny v nahodilém pořadí, napřeskáčku – i když jde stále případně o jednu osobu. Zde by měla existovat „pracovní schizofrenie“: soustředěním se vždy právě na jednu z těch rolí se osoba postupně „převtěluje“ z autora

do designéra a nakonec v sazeče, aby každou z těch fází vyřešila pokud možno bez negativního ovlivňování jinými fázemi. Tak lze dosáhnout maximálního efektu podobně jako dělbou práce mezi nezávislé odborníky.

Na první pohled je situace ve srovnání s (nedávnou) minulostí velmi obtížná. Má být jedinec zároveň autorem, úpravcem i sazečem? Není to mnoho odborností najednou? Když na to dříve byli školení specialisté, lze to dnes zvládnout všechno dohromady?

Ve skutečnosti se ale ani neptáme, zda je to možné, spíše předpokládáme, že je to již nezbytná nutnost. Ve srovnání s dřívějšími technologiemi máme navíc řadu podstatných výhod – co dříve trvalo řádově dny, zvládne počítač za vteřiny. Co se dříve dělalo pracně a ručně, dnes probíhá automaticky. Místo mnoha strojů a dalších pomůcek máme jen informaci v paměti počítače.

Oproti minulosti je však dnes zcela nezbytné, aby autor zvládl více věcí – chce-li být producentem dokumentu, nutně musí ovládnout taktéž práci úpravce a práci sazeče. Cílem celé koncepce tohoto kurzu je rozvíjení a podpora zejména těchto dvou rolí, a to tak, aby student získal schopnost veškerou práci provádět efektivně, s maximální podporou síly programového vybavení.

3.3 Typografické zásady

Role úpravce zůstává v téměř stejném rozsahu jako dříve a je to bezesporu největší mezera, kterou je potřeba patrně v každém kurzu zabývajícím se zpracováním dokumentů zaplnit. Všeobecné povědomí o některých zásadách sice pomalu roste, ale stále platí, že se jedná o oblast velmi málo pokrytou běžným vzdělávacím systémem a slabě vnímanou i běžnými uživateli dokumentů (čtenáři).

Ve většině kapitol je z těchto důvodů vždy uveden „typografický základ“ – jde o cíle, kterých chceme v dokumentu dosáhnout. Na to pak teprve navazuje technologická část, kde je řečeno, jak těchto cílů dosáhnout.

Jak je podle našich výukových zkušeností vhodné, jsou typografické zásady rozděleny přibližně na čtyři velké části – písmo a hladká sazba, smíšená a odstavcová sazba, stránky a jejich prvky, sestava dokumentu. Tyto části jsou rozmístěny do jednotlivých kapitol tak, aby na ně mohly navazovat příslušné technologické prvky. Kurz si nedělá ambice být rozsáhlou typografickou učebnicí – jednak takové učebnice existují (Beran et al., 2012; Kočička a Blažek, 2000), jednak by se takový objem nevešel do předpokládané časové dotace.

Osoba v roli úpravce se může alespoň částečně zorientovat v tom, jaké hlavní parametry určitých prvků se používají a jakých efektů lze dosáhnout.

3.4 Použitá technologie

Práci dřívějšího sazeče dnes reprezentuje schopnost využít vlastností a funkcí textových procesorů a dalších počítačových aplikací. Čím dokonaleji student ovládne některý ze systémů, tím efektivněji s ním bude umět pracovat a tím efektivněji bude vytvářet své dokumenty.

Hlavní součástí kurzu je z uvedených důvodů představení technologie a vhodných metodických postupů jejího využití. Koncepci „tří osob“ lze realizovat různými programovými systémy. Zcela obecně však platí, že těžištěm je využívání tzv. *strukturních značek* – technických nástrojů definujících význam jednotlivých prvků dokumentu. Autor jimi vkládá do dokumentu nezastupitelnou informaci, na kterou pak může navázat úpravce a sazeč.

Okrajově je zmíněna technologie strukturního značkování v textových procesorech typu Writer (podrobněji o tom pojednává Rybička, 2020), dále se však kurz věnuje využití systémů postavených na principu TEXu. Konkrétně je použit systém XELATEX, ale analogicky lze postupovat i v jiných podobných nadstavbách.

Důraz je opět kladen na strukturní značkování. Proto je oproti jiným učebnicím a návodným textům již od začátku věnován velký prostor tvorbě vlastních příkazů, které představují technickou realizaci strukturních značek. Tento přístup k tvorbě dokumentů by měl vést k daleko efektivnějšímu zpracování než postup, kdy se pro značkování používají pouze předdefinované (instantní) rekvizity. Tím ovšem nikterak nesnižujeme důležitost důkladného seznámení s předdefinovaným repetoárem, naopak – ten je používán jako „stavební materiál“ pro vlastní příkazy a v některých případech i jako inspirace, jak mohou strukturní značky vypadat.

Široké možnosti vytváření vlastních příkazů v systémech postavených na principu TEXu umožňují plně realizovat koncepci strukturních značek. Proto je velmi efektivní v tomto systému pracovat, a to zejména v oblasti odborných a vědeckých dokumentů. Nezanedbatelným cílem kurzu je pak také ukázat, že i přesto, že tzv. „běžné“ textové procesory (zejména včetně placených variant) jsou masivně rozšířeny, často zdaleka nepředstavují nejvhodnější volbu pro zpracování textů. Má pak smysl investovat určité úsilí do ovládnutí alternativy nacházející se navíc v oblasti open source softwaru.

4 Závěr

Celková koncepce kurzu „Nástroje open source pro zpracování textů“ vychází z dlouholetých pedagogických i praktických zkušeností se zpracováním textů. Očekávané přínosy této konstrukce jsou následující:

- sofistikovaný přístup k tvorbě dokumentů,
- zvládnutí technologie postavené na principu TEX,
- efektivní práce s dokumenty (při úpravách nebo znovupoužití),
- rozšíření možností práce s dokumenty oproti běžným textovým procesorům,
- zvýšení povědomí o oblasti open source.

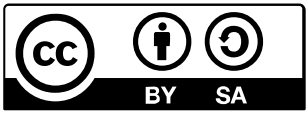
Při představení kurzu studentům v rámci Letní školy byly získány první reakce. I když se v omezené časové dotaci nebylo možné plně věnovat všem částem, byla zpětná vazba od studentů příznivá.

Literatura

1. Beran, V. et al. *Aktualizovaný typografický manuál.* Praha: Kafka design, 2012.
2. *ČSN 016910 Úprava dokumentů zpracovaných textovými procesory.* Praha: Ústav pro normalizaci a měření, 2014.
3. Kočička, P., Blažek, F. *Praktická typografie.* Praha: CP Books, 2000.
4. Rybička, J. *Zpracování textů v programu Word.* [online] [vid. 20. 11. 2021] Dostupné na `https://akela.mendelu.cz/\~{}rybicka/prez/zpract/odbprace/fastword2.docx`

Autor

Doc. Ing. Jiří Rybička, Dr., Ústav informatiky, Provozně ekonomická fakulta, Mendelova univerzita v Brně, Zemědělská 1, 613 00 Brno, e-mail: rybicka@mendelu.cz



1st International Conference on Open Source tools in Computer Science university education
Conference Proceedings

Author: Jiří Rybička (Ed.)
Mendel University in Brno

Publisher: Mendel University in Brno, Zemědělská 1, 613 00 Brno, Czech Republic
Graphic editing and typesetting: Jiří Rybička
Year of publishing: 2022
First edition
Number of pages: 102

ISBN 978-80-7509-898-6 (online ; pdf)
DOI https://doi.org/10.11118/978-80-7509-898-6