MASARYKOVA UNIVERZITA
Přírodovědecká fakulta

Karel ŠROT

NEKONEČNÉ ŘADY S PROGRAMEM MAPLE

Disertační práce

Školitel: RNDr. Roman Plch, Ph.D.

Brno, 2010

Bibliografická identifikace

Jméno a příjmení autora: Karel Šrot

Název disertační práce: Nekonečné řady s programem Maple

Název disertační práce anglicky: Infinite series with Maple

Studijní program: Matematika

Studijní obor: Obecné otázky matematiky

Školitel: Roman Plch

Rok obhajoby: 2010

Klíčová slova: nekonečné řady, mocninné řady, Fourierovy řady, Maple

Klíčová slova v angličtině: infinite series, power series, Fourier series, Maple

Abstrakt

Tato práce se zaměřuje na využití programu Maple při výuce nekonečných posloupností a řad reálných funkcí. V práci uvádíme konkrétní praktické příklady využívající program Maple, přičemž se soustředíme na čtyři hlavní témata – samotné *řešení problému*, kdy je program Maple prostředkem usnadňujícím rutinní a zdlouhavé výpočty, *automatizaci výpočtu* pomocí nových procedur, kritické *hodnocení správnosti výsledků* získaných prostřednictvím programu Maple a *vytváření grafických výstupů*, na kterých mimojiné demonstrujeme některé základní pojmy z tématu nekonečných řad.

První kapitola se věnuje posloupnostem funkcí, zejména limitě posloupnosti funkcí a stejnoměrné konvergenci. Na několika příkladech ilustrujeme význam těchto pojmů prostřednictvím série grafů a animací. Dále se věnujeme výpočtu těchto limit a ověřování stejnoměrné konvergence s využitím standardních prostředků programu Maple. Představíme také dvě nové procedury, které tyto výpočty automatizují. Na závěr se stručně zmíníme o vlastnostech stejnoměrně konvergentních posloupností.

Druhá kapitola je věnována nekonečným řadám funkcí. Zabývá se výpočtem oboru konvergence funkčních řad a to s využitím limitního podílového a odmocninového kritéria. Výhodou těchto kritérií je, že výpočet lze poměrně snadno automatizovat. Závěrem se věnujeme vytváření animací ilustrujících chování konvergentních funkčních řad uvnitř a vně oboru konvergence.

Třetí kapitola je věnována řadám mocninným. Pomocí prostředků programu Maple hledáme součty mocninných řad. Následně se zaměříme na problém opačný a hledáme Taylorovy a Maclaurinovy rozvoje daných funkcí. Zmíníme také standardní knihovnu `powseries`, která obsahuje procedury pro operace s mocninnými řadami a tyto procedury použijeme při řešení několika příkladů. Dále se věnujeme vytváření animací ilustrujících konvergenci Taylorových a Maclaurinových rozvojů. Závěrečná část kapitoly obsahuje několik příkladů demonstrujících využití mocninných řad při řešení diferenciálních rovnic.

Tématem čtvrté kapitoly jsou Fourierovy řady. Věnujeme se výpočtům rozvojů reálných funkcí do Fourierových trigonometrických řad a zabýváme se otázkou konvergence Fourierovy řady a tzv. *Gibbsovým jevem*. Podrobně představíme novou programovou knihovnu `FourierTrigSeries`, která slouží k výpočtům Fourierových řad a manipulaci s nimi. Procedury z této nové knihovny využijeme při výpočtech Fourierových řad několika funkcí a předvedeme si využití Fourierových řad při řešení diferenciálních rovnic. Představíme také internetovou prezentaci této knihovny s webovou aplikací pro výpočet Fourierových řad. Závěrečná část kapitoly se zabývá výpočty Fourierových řad vzhledem k vybraným typům ortogonálních polynomů.

Na přiloženém CD-ROM disku jsou k dispozici elektronické verze zápisníků s řešenými příklady a zdrojové kódy nových procedur a knihovny `FourierTrigSeries`.

Abstract

The thesis is concerned with the use of Maple in teaching infinite sequences and series of real functions. It contains concrete practical examples in Maple focussing on four main topics – the problem solving with Maple being used to facilitate the routine and tedious calculation, computing automation with new procedures, critical evaluation of the correctness of the results obtained by Maple, and generating output graphics used, among others, to demonstrate some basic concepts of infinite series.

Chapter one is concerned with function sequences, particularly with the limit and uniform convergence of a sequence. In several examples, these notions are demonstrated using a number of plots and animations. Calculating these limits and testing the uniform convergence by standard Maple tools are the next topics. Also two new procedures are introduced to automate such calculation. Finally, the properties of the uniform convergence of a sequence are mentioned.

Chapter two deals with infinite function series. This includes the determination of the domain of convergence of functional series using limit, ratio, and root tests. An advantage of such tests is that the calculation can easily be programmed. At the end, animations are generated demonstrating the behaviour of convergent functional series both inside and outside the convergence domain.

Chapter three is devoted to power series. Using Maple programming tools, the sums of power series are determined. Next the opposite problem is analysed, that is, finding the Taylor and Maclaurin expansions of given functions. Here also the standard `powseries` library is described offering procedures for operations with power series used to solve several particular problems. The next focus is on animations illustrating the convergence of Taylor and Maclaurin expansions. The final part of the chapter uses several examples to demonstrate the use of power series in finding solutions of differential equations.

The subject of chapter four is Fourier series. It deals with the calculation of expansions of real functions into Fourier trigonometric series including the question of the convergence of a Fourier series, and the Gibbs phenomenon. Special attention is paid to the new `FourierTrigSeries` program library used to calculate and manipulate Fourier series. The procedures contained in this library are used to determine the Fourier series of several functions and to demonstrate the application of Fourier series to finding solutions of differential equations. Also a web presentation of this library is included with a web application to calculate Fourier series. The final part of the chapter is concerned with the calculation of Fourier series with respect to selected types of orthogonal polynomials.

The attached CD contains electronic versions of worksheets with examples solved and the source codes of the new procedures and the `FourierTrigSeries` library.

Obsah

Úvod **2**

1 Posloupnosti funkcí **8**
1.1 Reprezentace posloupnosti funkcí 8
1.2 Bodová konvergence posloupnosti funkcí 10
1.3 Stejnoměrná konvergence posloupnosti funkcí 16

2 Řady funkcí **30**
2.1 Reprezentace řad funkcí 30
2.2 Výpočet oboru konvergence 31
2.3 Další metody ověřování konvergence funkčních řad 36
2.4 Ilustrování konvergence funkčních řad animací 37

3 Mocninné řady **41**
3.1 Obor konvergence mocninné řady 41
3.2 Součet a vlastnosti mocninných řad 44
3.3 Taylorovy a Maclaurinovy řady 49
3.4 Knihovna `powseries` 53
3.5 Animování rozvojů do mocninných řad 57
3.6 Užití mocninných řad 59

4 Fourierovy trigonometrické řady **68**
4.1 Výpočet metodou „krok za krokem“ 70
4.2 Konvergence Fourierovy řady 77
4.3 Použití Fourierových řad 79
4.4 Modul FourierTrigSeries 80
4.5 Řešené příklady 96
4.6 Fourierovy řady a jiné ortogonální systémy 106

Závěr **118**

Úvod

Počítače se již staly samozřejmou součástí našeho života a nachází uplatnění v široké škále oborů a lidských činností, vysoké školství nevyjímaje. Nezbytným předpokladem jejich úspěšného využití je existence vhodného programového vybavení. Z pohledu matematické analýzy se jedná zejména o tzv. *systémy počítačové algebry*[1], jež umožňují symbolickou manipulaci se zadanými výrazy v explicitním analytickém tvaru a provádění výpočtů bez nutnosti numerické aproximace.

Na vysokých školách se tyto systémy již prosadily a běžně jsou používány například při vytváření různých typů grafických výstupů, ať už se jedná o ilustrace do klasických skript, či digitální výstup využitelný v materiálech pro matematický e-learning. Studenti se s nimi však často setkávají pouze zprostředkovaně, méně časté je jejich využití „interaktivní formou“ přímo během výuky. Ale i tento přístup se již prosazuje. Konkrétně na Přírodovědecké fakultě Masarykovy univerzity v Brně jsou v posledních letech nabízeny posluchačům základních kurzů matematické analýzy volitelné, doplňující kurzy. Ty mají formu počítačového semináře a využívají řešení příkladů v systému Maple jako prostředek k hlubšímu pochopení látky probírané v kurzu základním.

Tato práce se zaměřuje na využití programu Maple při výuce nekonečných posloupností a řad reálných funkcí. Program Maple byl zvolen právě proto, že je již při výuce na Ústavu matematiky a statistiky PřF MU využíván. Kromě již zmiňovaných kurzů zde vznikly například výukové CD-ROMy [5] a [6], přičemž druhý uvedený je věnován také tématu nekonečných funkčních posloupností a řad. Tato práce si klade za cíl na tuto publikaci v jistém smyslu navázat, rozšířit spektrum řešených problémů a vytvořit nové procedury automatizující některé z výpočtů.

Cílem práce není zpracovat téma po teoretické stránce. Předpokládáme, že nezbytný teoretický základ student čerpá z jiných publikací, například [4], [7], [9] nebo [14], či přímo ze základních kurzů matematické analýzy. Naopak se zaměřujeme na využití programu Maple. Definice či tvrzení jsou v práci zmíněny pouze z důvodu názornosti, zejména v situacích,

[1] Computer Algebra systems, neboli CAS. Mezi nejznámější patří komerční programy Maple,[2] Mathematica,[3] Derive,[4] či volně dostupné programy Axiom[5] a Maxima.[6]

[2] Program Maple vznikl na University of Waterloo a je vyvíjen společností Maplesoft, `http://www.maplesoft.com`.

[3] Program Mathematica je vyvýjen společností Wolfram Research, `http://www.wolfram.com`.

[4] Program Derive byl vyvýjen společností Soft Warehouse, `http://www.derive-europe.com/`.

[5] `http://axiom-developer.org`

[6] `http://maxima.sourceforge.net`

kdy uvedené tvrzení tvoří podstatnou část následujících výpočtů.

V práce uvádíme konkrétní praktické příklady využívající program Maple, přičemž se soustředíme na následující témata.

Prvním je samotný postup při řešení problému. Program Maple je zde prostředkem usnadňujícím rutinní a zdlouhavé výpočty. Takto řešené příklady jsou vedeny metodou „krok za krokem“ a často kopírují postup užívaný při ručním výpočtu. Umožňují nám tak nezdržovat se časově náročnými mezivýpočty, jež jsou studentovi již dostatečně známé, ale naopak se více zaměřit na problém samotný. Tímto přístupem může vyučující probrat více příkladů rozličných typů a také příkladů s vyšší obtížností, jež by jinak byly pro studenty samostatně obtížně řešitelné.

Následujícím tématem je automatizace výpočtu. V několika případech jsme řešení problému automatizovali pomocí nových procedur. Ačkoliv ve velké míře kopírují dříve zmíněný přístup, nemusí být tyto procedury bežnému čtenáři srozumitelné a proto nejsou v práci rozebírány.[7] Smyslem těchto procedur je především usnadnit provedení příslušného výpočtu, ať již je výsledek použit pro kontrolu správnosti vlastního řešení nebo se jedná o součást řešení složitějšího problému. Dalším důvodem je také rozšíření programu Maple o chybějící funkcionalitu. To je zejména případ knihovny `FourierTrigSeries`, jež tvoří podstatnou část práce.

Se zmíněnými tématy úzce souvisí téma kritického myšlení. Má-li student problémy s využitím počítače nejen řešit, ale i úspěšně vyřešit, je nutné, aby k získaným výsledkům přistupoval kriticky a nepovažoval je automaticky za správné. V celé řadě byť i poměrně jednoduchých úloh získáme z programu Maple nesprávný výsledek. To může být způsobeno samotnými algoritmy programu Maple, přílišnou abstrakcí problému, ale i omezenými možnostmi prezentace těchto výsledků (zejména v případě grafických výstupů). V práci se snažíme na tyto situace upozorňovat. Student je tak veden ke kritickému zhodnocení obdržených výsledků a k ověření jejich správnosti prostřednictvím již nabytých znalostí. Získaná zkušenost se samozřejmě uplatní, ať již posuzuje správnost výsledků vlastních, tak výsledků obdržených prostřednictvím počítače.

Posledním tématem, na nejž se v práci zaměřujeme, je vytváření grafických výstupů. Počítačová grafika pomáhá u studentů rozvíjet geometrickou představivost. Formou grafů a animací lze například velmi názorně ilustrovat různé typy konvergence.[8] Dalším využitím může být vizuální kontrola správnosti získaných výsledků - konfrontování získaného řešení s obrazovým výstupem nás může upozornit na chybu v provedeném výpočtu.

Ze zdrojů dostupných v českém jazyce se tématu práce věnuje již zmíněný CD-ROM [6]. Mocninným řadám a programu Maple je věnována také práce [12]. Zabývá se zejména výpočtem Taylorových rozvojů, poloměru konvergence a ilustrací konvergence rozvoje prostřednictvím obrazové animace. Na Fourierovy trigonometrické řady je zaměřena i autorova předchozí práce [18].

[7] Důvodem je také skutečnost, že práce není koncipována jako učebnice programovaní v prostředí programu Maple.

[8] Ačkoliv i tato forma má své limity. Vhodná je zejména zabýváme-li se spojitými či po částech spojitými funkcemi.

Z anglicky psané literatury je třeba zmínit publikaci [1]. Obsahuje velké množství příkladů řešených pomocí programu Maple. Velmi populární jsou také knihy [11] a [15]. Věnují se širokému spektru témat z mnoha oblastí matematiky, bohužel příkladů opravdu řešených pomocí programu Maple obsahují poskromnu.

Nejrozsáhlejší kolekcí zdrojů pro program Maple je portál *Maple Application Center*.[9] Dostupné zápisníky se však velmi často tématicky překrývají či jsou prakticky totožné. Originální, či alespoň ty obsahově nejbohatší práce jsou uvedeny v seznamu literatury v části *Elektronické zdroje z kolekce MapleApps*.

Velmi oblíbené jsou v současné době také různé webové kalkulačky, umožňující provádět i poměrně komplikované výpočty v prostředí internetového prohlížeče. Mocninným řadám je věnována webová stránka vytvořená v rámci práce [12], jež slouží k počítání a animování grafů Taylorových rozvojů. Kalkulačka na výpočet Fourierových řad je součástí internetové prezentace knihovny `FourierTrigSeries` vzniklé v rámci této disertační práce.

Nyní krátce představíme obsah jednotlivých kapitol disertační práce.

Kapitola 1 – Posloupnosti funkcí

První kapitola se věnuje posloupnostem funkcí. Ačkoliv je práce věnovaná nekonečným řadám funkcí, na posloupnostech se lépe ilustrují některé základní pojmy. Také některé prováděné výpočty by byly v případě funkčních řad hůře proveditelné.

Nejdříve uvedeme několik možných způsobů reprezentace nekonečných posloupností funkcí v programu Maple. Následně se věnujeme limitě posloupnosti funkcí. Na několika příkladech ilustrujeme význam tohoto pojmu prostřednictvím série grafů a animací, dále se věnujeme výpočtu těchto limit s využitím standardních prostředků programu Maple, přičemž pozornost je věnována i situacím, kdy takový výpočet selhává. Tyto se pokoušíme automaticky řešit prostřednictvím nově vytvořené procedury `FindSequenceLimit`.

Poslední část kapitoly je věnována stejnoměrné konvergenci posloupností funkcí. Její podstatu opět ilustrujeme prostřednictvím grafů a animací. Dále se věnujeme ověřování stejnoměrné konvergence. Nejdříve přístupem „krok za krokem“ a následně automatizovanému výpočtu s využitím nové procedury `TestSequenceUniformConvergence`. Na závěr se stručně zmíníme i o vlastnostech stejnoměrně konvergentních posloupností, konkrétně derivování a integrování těchto posloupností člen po členu.

Kapitola 2 – Řady funkcí

Druhá kapitola je věnována nekonečným řadám funkcí. Podobně jako v kapitole první, nejdříve vysvětlíme způsob reprezentace funkčních řad v programu Maple, používaný v následujícím textu.

Dále se věnujeme výpočtu oboru konvergence funkčních řad. Výpočet opět provádíme „krok za krokem“ a to s využitím limitního podílového a odmocninového kritéria. Výho-

[9] `http://www.maplesoft.com/Applications/`

dou těchto kritérií je, že postup výpočtu lze lehce automatizovat, což také provádí nová procedura `TestSeriesConvergence`.

Na závěr se věnujeme vytváření animací ilustrujících chování konvergentních funkčních řad uvnitř a vně oboru konvergence.

Kapitola 3 – Mocninné řady

Třetí kapitola je věnována řadám mocninným. Nejdříve se vrátíme k výpočtu oboru konvergence, dále pomocí prostředků programu Maple hledáme také součet mocninné řady.

Následně se zaměříme na problém opačný a hledáme Taylorovy a Maclaurinovy rozvoje daných funkcí a to jak v tzv. zkráceném tvaru,[10] tak v uzavřeném tvaru. Zmíníme také standardní knihovnu `powseries`, která obsahuje procedury pro manipulaci s mocninnými řadami a tyto procedury použijeme při řešení několika příkladů.

V další části se věnujeme vytváření animací ilustrujících konvergenci Taylorových a Maclaurinových rozvojů. Závěrečná část obsahuje několik příkladů ilustrujících využití mocninných řad při aproximaci funkcí, přibližného výpočtu funkčních hodnot a při řešení diferenciálních rovnic metodou řad. Ačkoliv lze v těchto příkladech získat v programu Maple výsledek přímo, bylo naším cílem naopak ozřejmit podstatu těchto výpočtů.

Kapitola 4 – Fourierovy trigonometrické řady

Tématem poslední kapitoly jsou Fourierovy řady. Nejdříve se věnujeme výpočtům rozvojů reálných funkcí do Fourierových trigonometrických řad metodou „krok za krokem“. Dále se zabýváme otázkou konvergence Fourierovy řady a tzv. *Gibbsova jevu*.

Podrobně přestavíme novou programovou knihovnu `FourierTrigSeries`, která slouží k počítání Fourierových řad a manipulaci s nimi. Na jednoduchých příkladech předvedeme použití všech 17 procedur této knihovny. Zmíníme také webovou aplikaci pro výpočet Fourierových řad přímo prostřednictvím webového prohlížeče.

Následuje část s řešenými příklady, ve které procedury knihovny `FourierTrigSeries` využijeme při výpočtech Fourierových řad zadaných funkcí, věnujeme se opět Gibbsově efektu a jeho eliminaci metodou σ-aproximace a předvedeme využití Fourierových řad při řešení diferenciálních rovnic.

Záverečná část kapitoly se zabývá Fourierovými řadami vzhledem k ostatním ortogonálním systémům, konkrétně vzhledem k vybraným typům ortogonálních polynomů. Předvedeme využití programu Maple při hledání těchto polynomů a samozřejmě také při počítání Fourierových řad vzhledem k systémům těchto ortogonálních polynomů.

[10] Zkráceným tvarem rozumíme částečný součet mocninné řady plus výraz reprezentující příslušný zbytek.

Poznámky k použitým procedurám

V práci se často odvoláváme na procedury, které nejsou součástí standardní instalace programu Maple. Vetšina z nich byla vytvořena v rámci práce samotné, konkrétně se jedná o následující procedury.

`FindSequenceLimit` – Procedura počítá limitu posloupnosti funkcí, přičemž ověřuje hodnotu limity v bodech, kde standardní procedura `limit` selhává.

`TestSequenceUniformConvergence` – Procedura testuje stejnoměrnou konvergenci posloupnosti funkcí.

`plotStripAroundFunction` – Procedura s grafickým výstupem kreslící barevný pás dané šířky okolo zadané funkce. Procedura se používá při ilustraci stejnoměrné konvergence posloupnosti funkcí.

`barplot` – Procedura pro kreslení jednoduchého sloupcového grafu.

`TestSeriesConvergence` – Procedura počítá obor konvergence zadané funkční řady pomocí limitního podílového či odmocninového kritéria.

`plotVerticalStrip` – Procedura s grafickým výstupem kreslící vertikální barevný pás. Procedura se používá při ilustraci konvergence fukčních řad.

Dále v textu nejsou zmíněny pomocné procedury.

`convertToPiecewise` – Slouží k vytvoření definice po částech spojité funkce ze seznamu hraničních bodů a funkčních hodnot, jež tato funkce nabývá na intervalech mezi hraničními body.

`RealRangeUnion` – Procedura provede sjednocení předaných reálných intervalů.

`TestIfInRealRange` – Procedura ověřuje, zda daný bod patří do zadaného reálného intervalu či nikoliv.

Všechny uvedené procedury jsou dostupné v souboru *posl_a_rady_fci_proc.mpl*. Před jejich použitím v programu Maple je nutné načíst zdrojové kódy procedur pomocí příkazu `read`.[11]

Pro správnou funkčnost výše zmíněných procedur jsou nezbytné procedury `csum` a `inequalities`, které jsou součástí kolekce Maple Advisor Database [24], jejímž autorem je Robert Israel. Procedury jsou uloženy v souborech *csum5.txt* a *inequal5.txt* a tyto soubory musí být uloženy spolu se souborem *posl_a_rady_fci_proc.mpl* ve stejném adresáři.

[11] V zápisnících je pro tento účel používána sekvence příkazů
`currentdir("/home/karel/devel/maple/PhD/mws"): read "posl_a_rady_fci_proc.mpl":`

Konečně samotná knihovna `FourierTrigSeries` a v ní obsažené procedury byly vytvořeny jako součást disertační práce. Knihovna slouží k počítání Fourierových řad reálných funkcí a k manipulaci s nimi. Podrobně je popsána v samostatné sekci na straně 80.

Zbývající procedury zmiňované v textu jsou, často pro svoji jednoduchost, definovány přímo v místě použití. V opačném případě se jedná o procedury, jež jsou součástí standardní instalace programu Maple.

Zápisníky s řešenými příklady a zdrojové kódy použitých procedur jsou k dispozici na internetové stránce `http://www.math.muni.cz/~xsrot/nrspm` a také na přiloženém CD-ROM disku.

Kapitola 1

Posloupnosti funkcí

V této kapitole se budeme věnovat posloupnostem funkcí. Předvedeme si vytváření animací ilustrujících jejich konvergenci. Zmíníme nástoje, které Maple nabízí pro výpočet limity posloupnosti funkcí a také novou proceduru, která odstraňuje některé nedostatky standardní procedury `limit`. Podstatná část kapitoly je věnována stejnoměrně konvergentním posloupnostem funkcí a jejich vlastnostem.

1.1 Reprezentace posloupnosti funkcí

Pro snadnou manipulaci s posloupnostmi funkcí v programu Maple, a také s funkčními řadami, kterým se budeme věnovat v následujících kapitolách, je důležité zvolit vhodný způsob jejich reprezentace. V tomto výběru máme jistou volnost, ve velké míře závisí na vkusu a preferencích uživatele. Zvolená reprezentace by měla být přehledná a přitom by nám měla umožnit manipulaci jak se samotnou posloupností, tak i snadný přístup k jednotlivým členům posloupnosti. Níže uvádíme nejběžnější možnosti reprezentace posloupnosti funkcí.

a) Posloupnost reprezentujeme algebraickým výrazem.

```
> fn:=exp(n*x)/n!;
```

$$fn := \frac{\mathrm{e}^{(nx)}}{n!}$$

K jednotlivým členům posloupnosti pak přistupujeme pomocí příkazů `subs` nebo `eval`.

```
> subs(n=3,fn);
```

$$\frac{\mathrm{e}^{(3x)}}{3!}$$

případně

```
> eval(fn,n=3);
```

$$\frac{1}{6}\mathrm{e}^{(3x)}$$

Vzhledem ke způsobu, jakým přistupujeme k jednotlivým členům posloupnosti, není reprezentace posloupnosti funkcí algebraickým výrazem příliš vhodná. Často se však používá při volání procedur, které vyžadují jako parametr algebraický výraz. Význam neznámých vystupujících ve výrazu je pak upřesněn dalšími parametry, jež se proceduře předávají.

b) Posloupnost reprezentujeme funkcí[1] jedné proměnné vracející algebraický výraz.

```
> fn:=n->exp(n*x)/n!;
```

$$fn := n \to \frac{\mathrm{e}^{(nx)}}{n!}$$

K členům posloupnosti přistupujeme specifikováním argumentu této funkce.

```
> fn(3);
```

$$\frac{1}{6}\mathrm{e}^{(3x)}$$

c) Posloupnost reprezentujeme funkcí jedné proměnné vracející funkci.

```
> fn:=n->x->exp(n*x)/n!;
```

$$fn := n \to x \to \frac{\mathrm{e}^{(nx)}}{n!}$$

Členy posloupnosti jsou reprezentovány funkcemi a přistupujeme k nim opět specifikováním argumentu funkce.

```
> fn(3);
```

$$x \to \frac{\mathrm{e}^{(3x)}}{3!}$$

d) Posloupnost reprezentujeme funkcí dvou respektive více proměnných, přičemž první argument reprezentuje index posloupnosti a zbývající argument respektive argumenty představují proměnné jednotlivých funkcí (členů) v posloupnosti.

```
> fn:=(n,x)->exp(n*x)/n!;
```

$$fn := (n, x) \to \frac{\mathrm{e}^{(nx)}}{n!}$$

Členy posloupnosti jsou reprezentovány algebraickým výrazem a přistupujeme k nim specifikací zmíněných argumentů.

```
> fn(3,x);
```

$$\frac{1}{6}\mathrm{e}^{(3x)}$$

[1] Abychom v textu odlišili „matematické“ funkce od jiných funkcí definovaných v programu Maple, budeme druhé jmenované nazývat procedury. Výjimkou budou pouze procedury definované pomocí funkcionálního operátoru `->`, které jsou často používány právě pro reprezentaci funkcí.

V tomto textu budeme používat téměř výhradně poslední uvedený způsob, který umožňuje jednoduše přistupovat nejen k jednotlivým členům posloupnosti, ale zároveň také k číselné posloupnosti určené pevně zvoleným bodem x či přímo k funkční hodnotě konkrétní funkce. Ačkoliv v některých případech může být výhodnější zvolit jiný způsob reprezentace posloupnosti, v zájmu jednotnosti a srozumitelnosti se v dalším textu budeme tohoto způsobu i nadále držet.

1.2 Bodová konvergence posloupnosti funkcí

Dříve než se začneme zabývat hledáním limity posloupnosti funkcí, uveďme definici bodové konvergence pro posloupnost funkcí.

Definice. Nechť $\{f_n(x)\}_{n=1}^{\infty}$ je posloupnost funkcí definovaných na intervalu I a $x_0 \in I$ je libovolné. Je-li číselná posloupnost $\{f_n(x_0)\}_{n=1}^{\infty}$ konvergentní, říkáme, že posloupnost $\{f_n(x)\}_{n=1}^{\infty}$ je *konvergentní v bodě* x_0.

Řekneme, že posloupnost funkcí *bodově konverguje k funkci* $f(x)$ *na intervalu* I, jestliže konverguje v každém bodě $x \in I$, tj. ke každému $x \in I$ a každému $\varepsilon > 0$ existuje $n_0 \in \mathbb{N}$ tak, že pro všechna $n \in \mathbb{N}$, $n > n_0$, platí

$$|f_n(x) - f(x)| < \varepsilon.$$

Píšeme $\lim f_n(x) = f(x)$ pro $x \in I$ nebo $f_n \to f$ na I.

Bodová konvergence posloupnosti funkcí závisí na intervalu, na kterém konvergenci vyšetřujeme. Největší množinu (vzhledem k množinové inkluzi), na níž posloupnost funkcí bodově konverguje, nazýváme *oborem konvergence* posloupnosti funkcí $\{f_n(x)\}$.

V následující podkapitole se budeme věnovat také stejnoměrně konvergentním posloupnostem funkcí a pokusíme se ilustrovat rozdíly mezi konvergencí bodovou a stejnoměrnou.

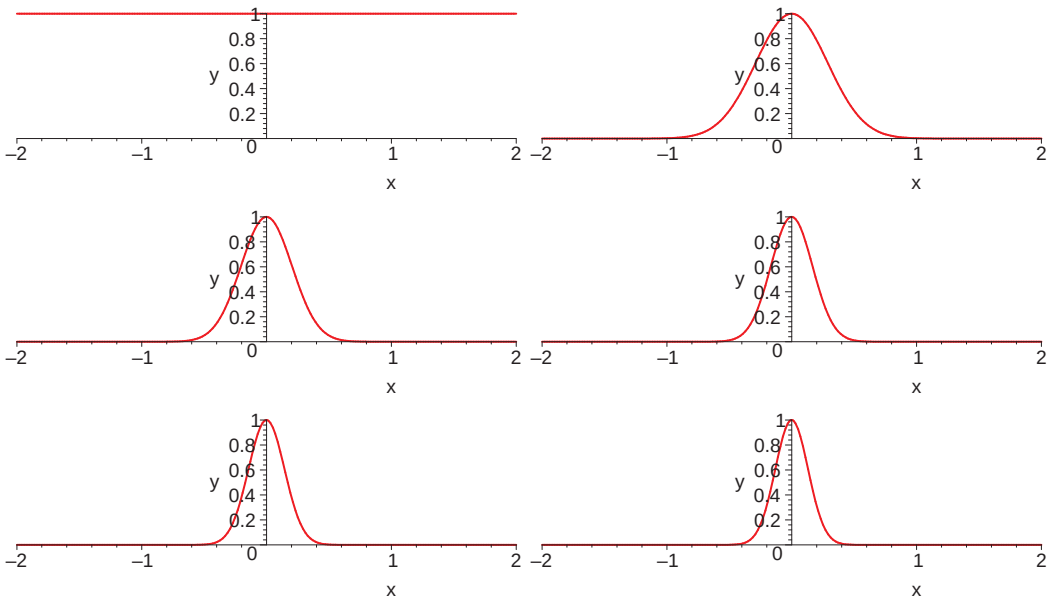
1.2.1 Znázornění posloupnosti funkcí grafem

Nepříliš sofistikované, ale dozajista přínosné, je využití programu Maple k ilustraci posloupnosti funkcí pomocí grafů nebo animace. To je výhodné například při hledání limity posloupnosti, zejména posloupnosti spojitých, či po částech spojitých, funkcí (pokud tato limita existuje). Jak si ukážeme v následující části, rutinním výpočtem často nezískáme přesnou odpověď. K vyřešení úlohy nám může pomoci právě vizuální kontrola, při které srovnáním získaného výsledku s grafy několika členů posloupnosti snadno zjistíme, zda (a proč) je získaný výsledek nesprávný.

Příklad 1.1. *Nakreslete grafy vybraných členů posloupnosti funkcí* $\{\mathrm{e}^{-nx^2}\}_{n=0}^{\infty}$.

Řešení. Při kreslení více grafů je výhodné definovat funkci pro jejich vykreslení.

```
> fn:=(n,x)->exp(-n*x^2):
```



Obr. 1.1: Vybrané náhledy z animace (grafy funkcí f_0, f_6, f_{12}, f_{18}, f_{24}, f_{29}).

```
> graf:=n->plot(fn(n,x), x=-2..2, scaling=constrained):
```

Graf funkce f_3 pak vykreslíme příkazem

```
> graf(3);
```

Více funkcí můžeme zobrazit v jednom grafu například příkazem

```
> plots[display](seq(graf(i),i=0..10));
```

Uvedením parametru `insequence=true` v proceduře `plots[display]` získáme animaci postupně zobrazující jednotlivé grafy.

```
> plots[display](seq(graf(i), i=0..29), insequence=true);
```

Vybrané náhledy z animace jsou na obrázku 1.1.

Při kreslení více funkcí do jednoho grafu je vhodné jednotlivé funkce barevně rozlišit. To lze provést například následující sérii příkazů:

```
> MAXCOLORS:=10: # nastaveni maximalniho poctu pouzitych barev
> graf:=n->plot(fn(n,x),x=-2..2,color=COLOR(HUE,min(0.7*n/MAXCOLORS,0.7))):
> plots[display](seq(graf(i),i=0..10));
```

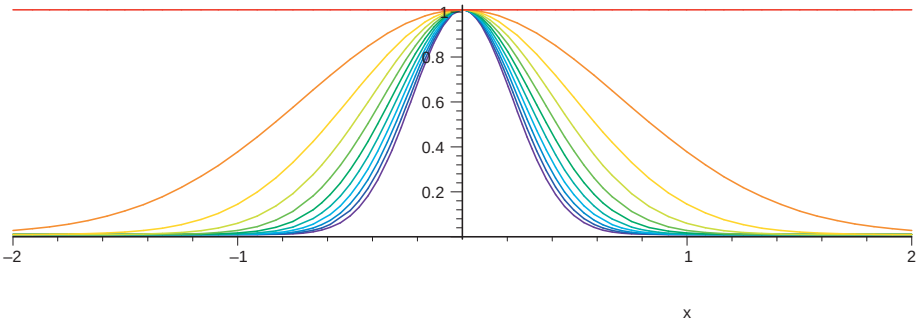
Jednotlivé funkce budou vykresleny spektrem barev, od červené po modrou. Získaný graf je na obrázku 1.2. □

Animace můžeme využít také v případě posloupnosti funkcí dvou proměnných.

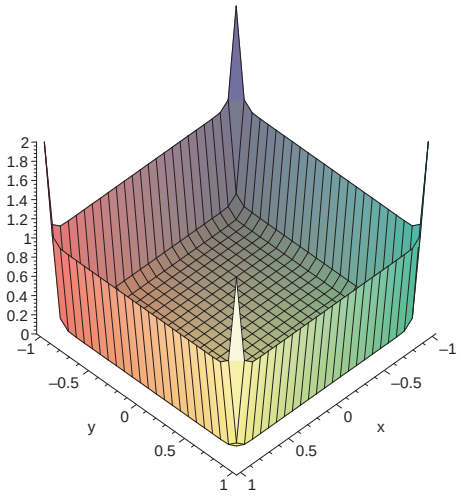
Příklad 1.2. *Pomocí animace najděte limitu posloupnosti funkcí* $\{|x|^n+|y|^n\}_{n=0}^{\infty}$ *na množině* $M=[-1,1]\times[-1,1]$.

Řešení.

```
> f:=(n,x,y)->abs(x)^n+abs(y)^n;
```



Obr. 1.2: Graf funkcí posloupnosti $\{e^{-nx^2}\}$ pro $n = 0, 1, \ldots, 10$.



Obr. 1.3: Graf funkce z posloupnosti při $n = 29$.

```
> plots[animate](plot3d,[f(n,x,y),x=-1..1,y=-1..1], n=0..29,frames=30,
  scaling=constrained,axes=framed);
```

Poslední snímek z animace je na obrázku 1.3. Na vnitřku množiny M je limita rovna 0, na hranici množiny rovna 1 s výjimkou bodů v „rozích“, kde je limita rovna 2. □

1.2.2 Výpočet limity posloupnosti funkcí

Program Maple můžeme samozřejmě použít pro výpočet limity posloupnosti funkcí. K tomuto účelu využijeme proceduru `limit`, která pro nalezení limity používá sofistikované algoritmy. V případě posloupnosti funkcí však další proměnná navíc (respektive proměnné u funkcí více proměnných) výpočet podstatně komplikuje. Proto je často nezbytné specifikovat, že se jedná o funkci reálné proměnné, případně i omezit její definiční obor. Ani to ale nemusí zajistit správnost výsledku. K tomu je tak vždy nutné přistupovat kriticky. Některá úskalí při výpočtu limit představíme na několika příkladech.

Příklad 1.3. *Najděte limitu posloupnosti funkcí* $\{\operatorname{arctg} nx\}_{n=0}^{\infty}$ *pro* $x \in \mathbb{R}$.

Řešení.

```
> fn:=(n,x)->arctan(n*x):
> limit(fn(n,x),n=infinity);
```

$$\frac{1}{2}csgn(x)\pi$$

Odpověď je správná, funkce `csgn` je rozšířením funkce *signum* na komplexní čísla. Jelikož se zabýváme funkcemi reálné proměnné, je vhodné toto specifikovat.[2]

```
> assume(x::real):
> limit(fn(n,x),n=infinity);
```

$$\frac{1}{2}signum(x)\pi$$

Limitou posloupnosti je tedy funkce $y = \frac{\pi}{2}\operatorname{sgn} x$. Všimněme si, že ačkoliv je posloupnost tvořena spojitými funkcemi, její limitou je funkce nespojitá v bodě $x = 0$. □

Příklad 1.4. *Najděte limitu posloupnosti funkcí* $\{\mathrm{e}^{-nx^2}\}_{n=0}^{\infty}$ *pro* $x \in \mathbb{R}$.

Řešení. Budeme postupovat jako u předchozího příkladu:

```
> fn:=(n,x)->exp(-n*x^2):
> assume(x::real):
> limit(fn(n,x),n=infinity);
```

$$0$$

Získaná odpověď však tentokrát není správná. Z předpisu posloupnosti nebo obrázků 1.1 a 1.2 je patrné, že v bodě $x = 0$ nabývají všechny funkce hodnoty 1. Můžeme to také ověřit výpočtem:

```
> limit(fn(n,0),n=infinity);
```

$$1$$

Ukazuje se, jak může graf několika členů posloupnosti pomoci[4] při ověření správnosti získaného výsledku a naopak, jak je důležité se na Maple slepě nespoléhat. □

Příklad 1.5. *Najděte limitu posloupnosti funkcí* $\{x^n\}_{n=0}^{\infty}$ *pro* $x \in [0, 1]$.

Řešení.

```
> fn:=(n,x)->x^n:
```

[2]Maple vkládá za název proměnné symbol $\sim$, pokud je blíže určen rozsah hodnot,[3] které může tato proměnná nabývat. Z důvodu přehlednosti nebudeme v textu tento symbol uvádět. V prostředí programu Maple toho lze docílit příkazem `interface(showassumed=0):`.

[3]Můžeme například specifikovat, že neznámá nabývá hodnot pouze z množiny přirozených čísel. Tuto informaci pak Maple využije třeba při úpravě algebraických výrazů, v nichž tato neznámá vystupuje. Více se lze dozvědět z nápovědy zadáním příkazů `?assume` či `?property`.

[4]Samozřejmě je to přínosem až u podstatně složitějších posloupností, kdy limita v bodě není patrná na první pohled.

```
> assume(x::RealRange(0,1)):
> limit(fn(n,x),n=infinity);
```

$$0$$

Odpověď opět není správná. V bodě $x = 1$ je limita posloupnosti rovna jedné.

Zde je vhodné zmínit ještě jedno překvapení, kterým nás může Maple zaskočit. Tím je neschopnost[5] určit limitu, pokud vyšetřovaný interval obsahuje kladné i záporné hodnoty. Zkoumaná posloupnost funkcí má limitu rovnu nule pro x z intervalu $(-1, 1)$, v bodě $x = 1$ je limita rovna jedné. Avšak Maple není schopen limitu určit ani na intervalu $[-\frac{1}{2}, \frac{1}{2}]$.

```
> assume(x::RealRange(-1/2,1/2)):
> limit(fn(n,x),n=infinity);
```

$$\lim_{n\to\infty} x^n$$

Pokud se omezíme na kladnou resp. zápornou část intervalu, jsou již limity určeny správně.

```
> assume(x::RealRange(-1/2,0)):
> limit(fn(n,x),n=infinity);
```

$$0$$

```
> assume(x::RealRange(0,1/2)):
> limit(fn(n,x),n=infinity);
```

$$0$$

□

Výpočet limity procedurou `FindSequenceLimit`

Procedura `FindSequenceLimit` slouží k výpočtu limity posloupnosti funkcí jedné proměnné. Nejdříve použije standardní proceduru `limit` k určení limity a následně získaný výsledek ověřuje v bodech, kde by procedura `limit` mohla selhat, tj. v krajních bodech intervalu, v bodech lokálních extrémů či v bodech nespojitosti.

Procedura očekává tři parametry. Prvním je algebraický výraz reprezentující posloupnost, druhým je proměnná reprezentující index a třetím proměnná vystupující jako neznámá jednotlivých funkcí v posloupnosti. Použití této procedury při řešení příkladu 1.4 by mohlo vypadat následovně:

```
> fn:=(n,x)->exp(-n*x^2):
> FindSequenceLimit(fn(n,x),n,x);
```

$$\begin{cases} 1 & x = 1 \\ 0 & otherwise \end{cases}$$

[5] Toto chování přetrvává až do aktuálně nejnovější verze Maple 11.

[6] Vzhledem ke struktuře parametrů funkce `FindSequenceLimit` a jednoduchosti řešeného příkladu je reprezentace posloupnosti pomocí funkce mírně kontraproduktivní. Ale v zájmu zachování jednotného stylu při řešení úloh se jí budeme držet i nadále.

Limitou posloupnosti je tedy funkce identicky rovna nule na celém $\mathbb{R}$ s výjimkou bodu $x = 0$, kde nabývá hodnotu 1.

Občas může být výhodné zobrazit podrobnější informace o průběhu výpočtu. Toho lze docílit úpravou hodnoty proměnné prostředí `infolevel`.[7]

Příklad 1.6. *Najděte limitu posloupnosti funkcí* $\{\frac{1}{nx}\}_{n=1}^{\infty}$ *pro* $x \in \mathbb{R} \setminus \{0\}$.

Řešení.

```
> infolevel[FindSequenceLimit]:=2:
> fn:=(n,x)->1/(n*x);
```

$$fn := (n, x) \rightarrow \frac{1}{nx}$$

```
> FindSequenceLimit(fn(n,x),n,x);
  FindSequenceLimit: Limit function found by Maple 0
  FindSequenceLimit: Derivative of the function -1/n/x^2
  FindSequenceLimit: Undefined points x = 0
  FindSequenceLimit: We will explore points x = -infinity, infinity
```

$$\begin{cases} undefined & x = 0 \\ 0 & otherwise \end{cases}$$

Hledanou limitou je funkce $y = 0$. □

U třetího předávaného parametru lze pomocí struktury `RealRange` blíže specifikovat interval, na kterém limitu posloupnosti funkcí hledáme. Níže je uvedeno řešení příkladu 1.5 s využitím procedury `FindSequenceLimit`.

```
> fn:=(n,x)->x^n:
> FindSequenceLimit(f(n,x),n,x=RealRange(0,1));
  FindSequenceLimit: Limit function found by Maple 0
  FindSequenceLimit: Derivative of the function x^(n-1)*n
  FindSequenceLimit: We will explore points x = 0, 1
```

$$\begin{cases} 1 & x = 1 \\ 0 & otherwise \end{cases}$$

Chceme-li ve struktuře `RealRange` vyjmout z intevalu krajní bod (v případě otevřených nebo polootevřených intervalů), použijeme rezervované slovo `Open`.[8]

```
> FindSequenceLimit(f(n,x),n,x=RealRange(0,Open(1));
  FindSequenceLimit: Limit function found by Maple 0
  FindSequenceLimit: Derivative of the function x^(n-1)*n
  FindSequenceLimit: We will explore points x = 0
```

$$0$$

[7] Více informací získáme zadáním příkazu `?userinfo`.

[8] Bližší informace o specifikaci intervalů lze získat z nápovědy zadáním příkazu `?realrange`.

Procedura si dokáže poradit také s posloupnostmi některých po částech spojitých funkcí.

Příklad 1.7. *Najděte limitu posloupnosti funkcí* $\{f_n(x)\}_{n=1}^{\infty}$ *zadaných předpisem*

$$f_n(x) = \begin{cases} \frac{1}{n} & x < 0 \\ x^n & x \leq 1 \\ x & x < 2 \\ \frac{1}{2n} & otherwise \end{cases}$$

Řešení.

```
> fn:=(n,x)->piecewise(x<0,1/n,x<=1,x^n,x<2, x,1/(2*n)):
> FindSequenceLimit(fn(n,x),n,x);
  FindSequenceLimit: Limit function found by Maple piecewise(x<1, 0, x<2, x, 2<=x, 0)
  FindSequenceLimit: Derivative of the function piecewise(x<0, 0, x<1, x^(n-1)*n, x<2,
  1, 2<=x, 0)
  FindSequenceLimit: We will explore points x = -infinity, 0, 1, 2, infinity
```

$$\begin{cases} 0 & x < 1 \\ 1 & x = 1 \\ x & x < 2 \\ 0 & 2 \leq x \end{cases}$$

□

1.3 Stejnoměrná konvergence posloupnosti funkcí

Stejnoměrná konvergence hraje v teorii nekonečných posloupností a řad funkcí velmi důležitou roli, protože nám umožňuje[9] například zaměnit pořadí sumace a integrace a integrovat tak řadu člen po členu (nebo naopak).

Definice. Řekneme, že posloupnost funkcí $\{f_n(x)\}_{n=1}^{\infty}$ konverguje stejnoměrně k funkci $f(x)$ na intervalu I, jestliže ke každému $\varepsilon > 0$ existuje $n_0 \in \mathbb{N}$ tak, že pro všechna $n \in \mathbb{N}$, $n > n_0$, a všechna $x \in I$ platí

$$|f_n(x) - f(x)| < \varepsilon.$$

Píšeme $f_n \Rightarrow f$ na I.

Následně si předvedeme využití programu Maple při ilustrování stejnoměrné konvergence pomocí animací a také při ověřování stejnoměrné konvergence přímým výpočtem.

[9] Stejnoměrná konvergence je samozřejmně často jen jednou z nutných podmínek.

1.3.1 Ilustrování stejnoměrné konvergence animací

Pomocí animací lze u spojitých funkcí (resp. u funkcí spojitých po částech) stejnoměrnou konvergenci velmi srozumitelně demonstrovat. To ale není jediný jejich přínos. Podobně jako u hledání limity posloupnosti funkcí, také při vyšetřování stejnoměrné konvergence nám mohou grafy respektive animace posloupnosti mnohé napovědět.

V odstavci 1.2.1 jsme zmínili snadný způsob vytváření animací. V následujícím příkladu si představíme názorný způsob ilustrace stejnoměrné konvergence posloupnosti. Naším cílem bude pomocí animace předvést, že počínaje jistou hodnotou indexu n_0 leží všechny funkce $f_n(x)$, $n > n_0$ v libovolně malém okolí limity posloupnosti, funkce $f(x)$. Nebudeme zde ověřovat stejnoměrnou konvergenci posloupnosti, tomuto tématu se budeme věnovat v následující části.

Příklad 1.8. *Pomocí animace ilustrujte stejnoměrnou konvergenci posloupnosti funkcí* $\{x^n - x^{n+1}\}_{n=1}^{\infty}$ *na intervalu* $[0, 1]$.

Řešení. Nejdříve je vhodné zjistit limitu zadané posloupnosti. K tomu můžeme využít některý ze způsobů uvedených v předchozích odstavcích.

```
> fn:=(n,x)->x^n-x^(n+1):
> FindSequenceLimit(fn(n,x),n,x=RealRange(0,1));
```

$$0$$

Limitou posloupnosti je[10] tedy funkce $y = 0$. Pro vykreslení pásu v okolí limitní funkce použijeme proceduru `plotStripAroundFunction`. Tato proceduru očekává tři parametry. Prvním je požadovaná funkce, druhým je neznámá s uvedeným rozsahem hodnot a třetím šířka pásu. Pomocí volby `color` lze nastavit barvu pásu, implicitně je touto barvou zelená. Následující příkaz uloží do proměnné `graf_pasu` graf žlutého pásu o poloměru 0,1.

```
> graf_pasu:=plotStripAroundFunction(0,x=0..1,0.1,color=yellow):
```

Funkci pro kreslení jednotlivých grafů definujeme podobně jako v příkladu 1.1. Pouze jej doplníme o graf limitní funkce obklopené barevným pásem.

```
> graf_limity:=plot(0,x=0..1,color=green):
> graf:=n->plots[display](plot(fn(n,x),x=0..1),graf_limity,graf_pasu):
> plots[display](seq(graf(i),i=1..10),insequence=true,scaling=constrained);
```

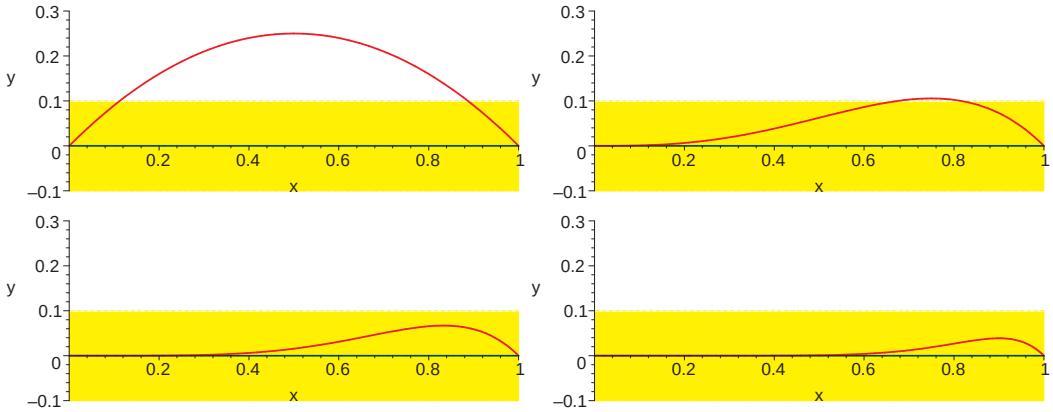
Vybrané náhledy[11] z animace jsou na obrázku 1.4.

Následně můžeme šířku pásu a počet grafů v animaci měnit a demonstrovat tak princip stejnoměrné konvergence. □

Příklad 1.9. *Ilustrujte animací stejnoměrnou konvergenci posloupnosti funkcí* $\{\frac{nx}{1+n+x}\}_{n=1}^{\infty}$ *na intervalu* $[0, 1]$.

[10] Samozřejmě se nezapomeneme nad správností výsledku zamyslet.

[11] V některým verzích programu Maple mohou být grafy funkcí barevným pásem překryty. V tomto případě pomůže změnit pořadí vykreslovaných grafů v definici funkce `graf`.



Obr. 1.4: Vybrané náhledy z animace (grafy funkcí f_1, f_3, f_5, f_9).

Řešení. Postupujeme jako v předchozím příkladě. Nejdříve tedy určíme limitu posloupnosti a poté sestrojíme animaci.

```
> fn:=(n,x)->(n*x)/(1+n+x):
> FindSequenceLimit(fn(n,x),n,x=RealRange(0,1));
```

$$x$$

```
> graf_pasu:=plotStripAroundFunction(x,x=0..1,0.1,color=yellow):
> graf_limity:=plot(x,x=0..1,color=green):
> graf:=n->plots[display](plot(fn(n,x),x=0..1),graf_limity,graf_pasu):
> plots[display](seq(graf(i),i=1..50),insequence=true,
  scaling=constrained);
```

Vybrané náhledy z animace jsou na obrázku 1.5. □

1.3.2 Ověřování stejnoměrné konvergence

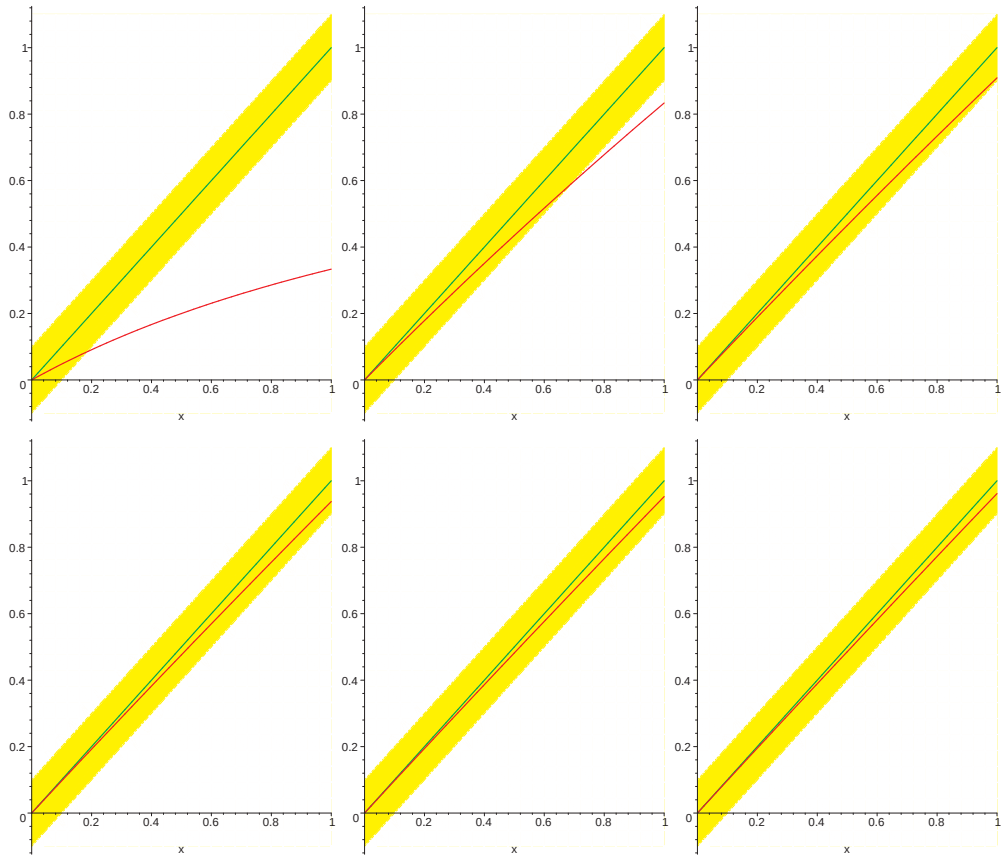
V následujícím textu si představíme kritérium stejnoměrné konvergence pro posloupnost funkcí. Jedná se prakticky o přepis definice s využitím tzv. *suprémové metriky*. Výhodou tohoto kritéria je jeho jednoduchost, která umožňuje snadnou automatizaci výpočtu programem Maple.

Věta. Nechť $\{f_n(x)\}_{n=1}^{\infty}$ je konvergentní posloupnost funkcí na intervalu I, $f(x)$ její limita. Označme

$$a_n = \sup\{|f_n(x) - f(x)|; x \in I\}$$

Pak posloupnost $\{f_n(x)\}_{n=1}^{\infty}$ konverguje stejnoměrně k $f(x)$ právě tehdy, když $\lim a_n = 0$.

Pro naše účely toto kritérium lehce upravíme a to tak, že namísto supréma budeme zkoumat lokální extrémy rozdílu $f_n(x) - f(x)$. Pokud při $n \to \infty$ budou limity funkčních



Obr. 1.5: Vybrané náhledy z animace (grafy funkcí f_1, f_{10}, f_{20}, f_{30}, f_{40}, f_{50}).

hodnot těchto extrémů (na intervalu I) rovny nule, bude i $\lim a_n = 0$ a tedy posloupnost konverguje stejnoměrně. Naopak, pokud je některá z těchto limit nenulová, o stejnoměrnou konvergenci se nejedná.

Příklad 1.10. *Vyšetřete stejnoměrnou konvergenci posloupnosti funkcí* $\{x^n - x^{n+1}\}_{n=1}^{\infty}$ *na intervalu* $[0, 1]$.

Řešení.

```
> fn:=(n,x)->x^n-x^(n+1):
```

Jedná se o posloupnost z příkladu 1.8, její limitou je funkce $y = 0$. Posloupnost tak zároveň představuje posloupnost rozdílů $f_n(x) - f(x)$.

```
> lim_fn:=x->0:
> rozdil[12]:=(n,x)->fn(n,x)-lim_fn(x):
```

Nyní budeme hledat body, ve kterých tyto rozdíly nabývají lokálních extrémů. Určíme první derivaci uvažovaných rozdílů a najdeme body, kdy je první derivace rovna nule nebo neexistuje.

```
> diff(rozdil(n,x),x);
```

$$\frac{x^n n}{x} - \frac{x^{(n+1)}(n+1)}{x}$$

```
> extr:=solve(%=0,x);
```

$$extr := \frac{n}{n+1}$$

Lokálních extrémů tedy rozdíly mohou nabývat v bodě $x = \frac{n}{n+1}$, dále v krajních bodech intervalu, tj. bodech $x = 0$ a $x = 1$. Určíme limity v těchto bodech pro $n \to \infty$. Aby byla zkoumaná posloupnost stejnoměrně konvergentní, musí být tyto limity rovny nule. Naopak, pokud některá z těchto limit bude nenulová, posloupnost stejnoměrně konvergentní není.

```
> limit(rozdil(n,extr),n=infinity);
```

$$0$$

```
> limit(rozdil(n,0),n=infinity);
```

$$0$$

```
> limit(rozdil(n,1),n=infinity);
```

$$0$$

Vyšetřovaná posloupnost funkcí tedy stejnoměrně konverguje k funkci $y = 0$. □

Příklad 1.11. *Vyšetřete stejnoměrnou konvergenci posloupnosti* $\{x^n - x^{2n}\}_{n=1}^{\infty}$ *na intervalu* $[0, 1]$.

[12] Zavedení proměnných `lim_fn` a `rozdil` se může zdát v tomto případě zbytečné, avšak má význam v případě, kdy limita není identicky rovna nule. Zároveň lépe ilustrují postup výpočtu.

Řešení.

```
> fn:=(n,x)->x^n-x^(2*n):
```

Jedná se o posloupnost podobnou posloupnosti z předchozího příkladu, také v tomto případě je limitou posloupnosti funkce $y = 0$. Při výpočtu budeme postupovat jako v předchozím příkladě.

```
> lim_fn:=x->0:
> rozdil:=(n,x)->fn(n,x)-lim_fn(x):
> diff(rozdil(n,x),x);
```

$$\frac{x^n n}{x} - \frac{2x^{(2n)} n}{x}$$

```
> extr:=solve(%=0,x);
```

$$extr := \mathrm{e}^{\left(-\frac{\ln(2)}{n}\right)}$$

Prozkoumáme body podezřelé na výskyt lokálního extrému.

```
> limit(rozdil(n,extr),n=infinity);
```

$$\frac{1}{4}$$

```
> limit(rozdil(n,0),n=infinity);
```

$$0$$

```
> limit(rozdil(n,1),n=infinity);
```

$$0$$

Jelikož v bodě $x = \mathrm{e}^{-\frac{\ln 2}{n}}$ je hodnota limity pro $n \to \infty$ různá od nuly, není vyšetřovaná posloupnost stejnoměrně konvergentní.

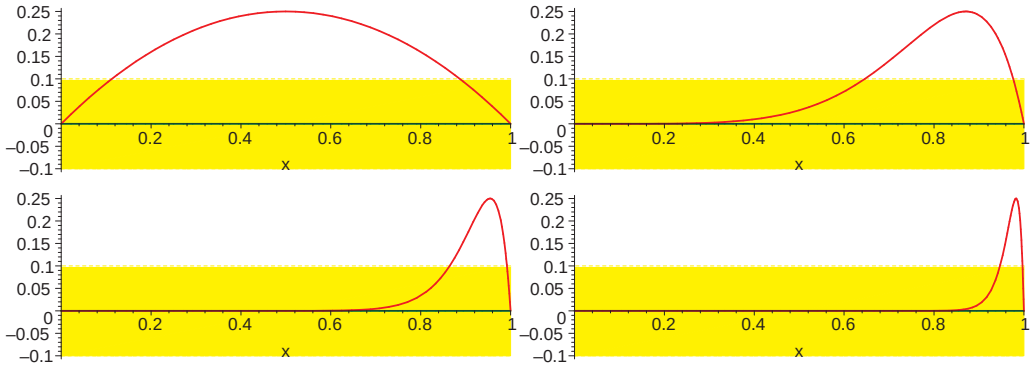
Uvedená posloupnost je tedy příkladem bodově, avšak ne stejnoměrně, konvergentní posloupnosti funkcí. V takovém případě se často jedná o posloupnost spojitých funkcí, jejíž limitou je funkce nespojitá. Tato posloupnost funkcí je ale zajímavá tím, že stejně jako její členy, také limita je funkce spojitá. Zůstaňme proto ještě chvíli u této posloupnosti a sestrojme animaci postupem uvedeným v předchozí části.

```
> graf_pasu:=plotStripAroundFunction(0,x=0..1,0.1,color=yellow):
> graf_limity:=plot(0,x=0..1,color=green):
> graf:=n->plots[display](plot(fn(n,x),x=0..1),graf_limity,graf_pasu):
> plots[display](seq(graf(i),i=1..40),insequence=true,
   scaling=constrained);
```

Vybrané náhledy z animace jsou na obrázku 1.6. □

Příklad 1.12. *Vyšetřete stejnoměrnou konvergenci posloupnosti* $\{\frac{xn}{1+n+x}\}_{n=1}^{\infty}$ *na intervalu* $[0, 1]$.

Řešení.



Obr. 1.6: Vybrané náhledy z animace (grafy funkcí f_1, f_5, f_{15} a f_{40}).

```
> fn:=(n,x)->x*n/(1+n+x):
```

Určíme limitu posloupnosti.

```
> limit(fn(n,x),n=infinity);
```

$$x$$

Limitou posloupnosti je funkce $y = x$.

```
> lim_fn:=x->x:
> rozdil:=(n,x)->fn(n,x)-lim_fn(x):
> diff(rozdil(n,x),x);
```

$$\frac{n}{1+n+x} - \frac{xn}{(1+n+x)^2} - 1$$

```
> extr:=solve(%=0,x);
```

$$extr := -n-1+\sqrt{n^2+n}, -n-1-\sqrt{n^2+n}$$

Tentokrát jsme získali dva body, ve kterých je první derivace rovna nule. Na první pohled není patrné, zda limity pro $n \to \infty$ patří do vyšetřovaného intervalu $[0, 1]$. Proto tyto limity určíme.

```
> limit(extr[1]),n=infinity);
```

$$-\frac{1}{2}$$

```
> limit(extr[2],n=infinity);
```

$$-\infty$$

Obě limity leží vně intervalu $[0, 1]$. Skoro všechny[13] funkce v posloupnosti tak nemají lokální extrém uvnitř intervalu $[0, 1]$. Proto dále ověřujeme již jen krajní body intervalu.

```
> limit(rozdil(n,0),n=infinity);
```

$$0$$

[13] Tedy pouze pro konečný počet funkcí v posloupnosti toto tvrzení neplatí.

```
> limit(rozdil(n,1),n=infinity);
```

$$0$$

Obě limity jsou rovny nule, vyšetřovaná posloupnost tedy stejnoměrně konverguje k funkci $y = x$. □

Procedura TestSequenceUniformConvergence

Metodu stručně prezentovanou v předchozích třech příkladech lze poměrně úspěšně automatizovat. Procedura `TestSequenceUniformConvergence` ověřuje stejnoměrnou konvergenci posloupnosti funkcí jedné proměnné výše uvedenou metodou.

Procedura má tři povinné argumenty. Prvním je vyšetřovaná posloupnost, druhým proměnná reprezentující v posloupnosti index, třetím proměnná jednotlivých funkcí (případně se specifikovaným definičním oborem). Čtvrtým, nepovinným, argumentem je limita posloupnosti. Pokud není parametr uveden, pokusí se procedura limitu posloupnosti zjistit pomocí procedury `limit`.

Použití procedury si předvedeme na několika příkladech. Také v tomto případě je výhodné povolit zobrazení podrobných informací o probíhajících výpočtech nastavením proměnné prostředí `infolevel` přikazem

```
> infolevel[TestSequenceUniformConvergence]:=2:
```

Příklad 1.13. *Vyšetřete stejnoměrnou konvergenci posloupnosti* $\{\sqrt{x^2 + \frac{1}{n}}\}_{n=1}^{\infty}$ *na intervalu* $[0, 1]$.

Řešení.

```
> fn:=(n,x)->sqrt(x^2+(1/n)):
> TestSequenceUniformConvergence(fn(n,x), n, x=0..1);
  TestSequenceUniformConvergence: Using the limit function: x
  TestSequenceUniformConvergence: Error function: (x^2+1/n)^(1/2)-x
  TestSequenceUniformConvergence: Extrema points: [0, 1]
  TestSequenceUniformConvergence: Error value at extrema points: [(1/n)^(1/2),
  (1+1/n)^(1/2)-1]
  TestSequenceUniformConvergence: Maximum limit value at extrema points: 0
```

$$true$$

Rozeberme si blíže význam jednotlivých řádků. Protože jsme explicitně neuvedli limitu posloupnosti funkcí, byla limita určena pomocí procedury `limit`. Z výpisu je patrné, že získanou limitou je funkce $y = x$. Na druhem řádku je uveden rozdíl $f_n(x) - f(x)$, rozdíl funkce z posloupnosti a uvažované limity. Na třetím seznam bodů, které budou testovány na výskyt lokálních extrémů (v seznamu již nejsou obsaženy body, ve kterých funkce může nabývat lokálních extrému, ale samy leží mimo vyšetřovaný interval), na čtvrtém hodnoty rozdílu v těchto bodech. Poslední řádek informačního výpisu uvádí maximum z limit absolutních hodnot rozdílů pro $n \to \infty$.

Procedura vrátila hodnotu *true*, posloupnost tedy na intervalu $[0, 1]$ stejnoměrně konverguje k funkci $y = x$. □

Příklad 1.14. *Vyšetřete stejnoměrnou konvergenci posloupnosti* $\{e^{n(x-1)}\}_{n=1}^{\infty}$ *na intervalu* $[-\infty, \frac{9}{10}]$.

Řešení. Budeme postupovat jako u předchozího příkladu.

```
> fn:=(n,x)->exp(n*(x-1)):
> TestSequenceUniformConvergence(fn(n,x), n, x=-infinity..0.9);
  TestSequenceUniformConvergence: Could not find the limit function
```

$$FAIL$$

Procedurou `limit` se nepodařilo určit limitu posloupnosti. Například pomocí grafů několika funkcí z posloupnosti můžeme nabýt podezření, že limitou je funkce $y = 0$. Proto tuto limitu v proceduře explicitně uvedeme na pozici čtvrtého parametru.

```
> TestSequenceUniformConvergence(fn(n,x), n, x=-infinity..0.9, 0);
  TestSequenceUniformConvergence: Using the limit function: 0
  TestSequenceUniformConvergence: Error function: exp(n*(x-1))
  TestSequenceUniformConvergence: Extrema points: [-infinity, .9]
  TestSequenceUniformConvergence: Error value at extrema points: [exp(-n*infinity),
  exp(-.1*n)]
  TestSequenceUniformConvergence: Maximum limit value at extrema points: 0
```

$$true$$

Posloupnost funkcí tedy konverguje stejnoměrně k funkci $y = 0$. Prozkoumejme dále interval $[-\infty, 1]$:

```
> TestSequenceUniformConvergence(fn(n,x), n, x=-infinity..1, 0);
  TestSequenceUniformConvergence: Using the limit function: 0
  TestSequenceUniformConvergence: Error function: exp(n*(x-1))
  TestSequenceUniformConvergence: Extrema points: [-infinity, 1]
  TestSequenceUniformConvergence: Error value at extrema points: [exp(-n*infinity),
  1]
  TestSequenceUniformConvergence: Maximum limit value at extrema points: 1
```

$$false$$

Na intervalu $[-\infty, 1]$ již posloupnost nekonverguje stejnoměrně. To je způsobeno tím, že ačkoliv jsou jednotlivé funkce z posloupnosti na tomto intervalu spojité, limitou je funkce nespojitá v bodě $x = 1$. □

V posledním příkladu jsme si ukázali, že pokud posloupnost spojitých funkcí konverguje k funkci nespojité, nemůže být tato konvergence stejnoměrná. Naopak platí, že posloupnost nespojitých funkcí může konvergovat stejnoměrně k funkci spojité. V následujícím příkladu je jedna taková triviální posloupnost uvedena.

Příklad 1.15. *Vyšetřete stejnoměrnou konvergenci posloupnosti* $\{f_n(x)\}_{n=1}^{\infty}$ *pro* $x \in \mathbb{R}$, *přičemž funkce* $f_n(x)$ *jsou dány předpisem*

$$f_n(x) = \begin{cases} \frac{1}{n} & x < 0 \\ \frac{2}{n} & x \geq 0 \end{cases}.$$

Řešení. Pokud ověřujeme stejnoměrnou konvergenci na celém $\mathbb{R}$, není nutné u třetího parametru uvádět definiční obor.

```
> fn:=(n,x)->(n*x)/(1+n^2*x^2):
> TestSequenceUniformConvergence(fn(n,x), n, x);
  TestSequenceUniformConvergence: Using the limit function: 0
  TestSequenceUniformConvergence: Error function: piecewise(x < 0,1/n,0 <= x,2/n)
  TestSequenceUniformConvergence: Derivative not defined at points: [0]
  TestSequenceUniformConvergence: Extrema points: [-infinity, infinity, 0]
  TestSequenceUniformConvergence: Error value at extrema points: [1/n, 2/n, 2/n]
  TestSequenceUniformConvergence: Maximum limit value at extrema points: 0
```

$$true$$

Posloupnost funkcí tedy konverguje stejnoměrně k funkci $y = 0$ na celém $\mathbb{R}$. □

K vyšetřování stejnoměrné konvergence se ještě vrátíme v následující kapitole věnované nekonečným řadám funkcí.

1.3.3 Vlastnosti stejnoměrně konvergentních posloupností

Integrování posloupnosti člen po členu

Jak již bylo řečeno v úvodu, stejnoměrná konvergence posloupnosti nám umožňuje zaměnit pořadí výpočtu limity posloupnosti a integrace. Přesněji to vyjadřuje následující věta:

Věta. Nechť posloupnost funkcí $\{f_n(x)\}$ stejnoměrně konverguje na intervalu $[a, b]$ k funkci f. Jsou-li všechny funkce $f_n(x)$ integrovatelné na $[a, b]$, je i $f_n(x)$ integrovatelná na $[a, b]$ a platí $\int_a^b f(x)\,\mathrm{d}x = \lim \int_a^b f_n(x)\,\mathrm{d}x$, tj.

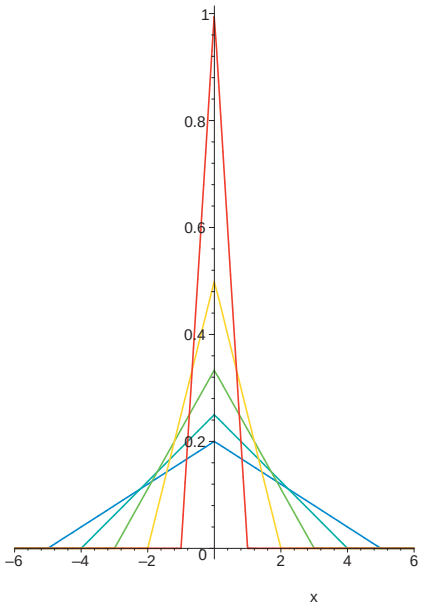
$$\int_a^b \left(\lim_{n\to\infty} f_n(x)\right)\,\mathrm{d}x = \lim_{n\to\infty} \int_a^b f_n(x)\,\mathrm{d}x.$$

Tvrzení věty budeme ilustrovat na jednoduchém příkladě.

Příklad 1.16. *Vyšetřete konvergenci pousloupnosti funkcí* $\{f_n(x)\}_{n=1}^{\infty}$ *zadaných předpisem*

$$f_n(x) = \begin{cases} \frac{x+n}{n^2} & x \in [-n, 0) \\ \frac{n-x}{n^2} & x \in [0, n] \\ 0 & jinak \end{cases}$$

na oboru $\mathbb{R}$ *a najděte limitu posloupnosti* $\left\{\int_a^b f_n(x)\,\mathrm{d}x\right\}_{n=1}^{\infty}$, *kde* $a, b \in \mathbb{R}$.



Obr. 1.7: Grafy funkcí $\{f_n\}$, $n = 1, 2, \ldots, 5$.

Řešení.

```
> fn:=(n,x)->piecewise(x<-n,0,x<0,(x+n)/n^2,x<n,(n-x)/n^2,0):
```

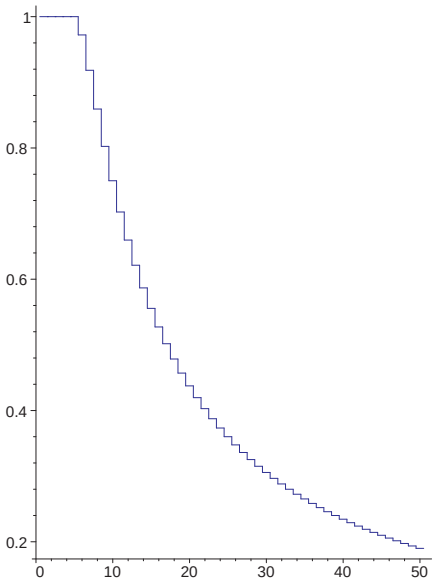
Abychom získali představu o podobě funkcí v posloupnosti, vykreslíme grafy několika z nich. Grafy jednotlivých funkcí barevně rozlišíme (přechodem od červené po modrou) způsobem použitým v příkladu 1.1.

```
> MAXCOLORS:=5:
> graf:=n->plot(fn(n,x),x=-6..6,thickness=2,color=COLOR(HUE,
   min(0.7*(n-1)/MAXCOLORS,0.7))):
> plots[display](seq(graf(i),i=1..5));
```

Výstup posledního příkazu je na obrázku 1.7. Na intervalu $[-n, n]$ tvoří graf funkce odvěsny rovnoramenného trojúhelníka s vrcholy $[-n, 0]$, $[0, \frac{1}{n}]$ a $[n, 0]$, vně tohoto intervalu je funkční hodnota rovna nule. Z toho je patrné, že posloupnosti konverguje stejnoměrně k funkci $y = 0$.

Nyní se zabývejme číselnou posloupností $\left\{\int_{-5}^{5} f_n(x)\,\mathrm{d}x\right\}_{n=1}^{\infty}$. Tato posloupnost splňuje podmínky předchozí věty, proto je limita této posloupnosti rovna nule. S tím bychom se mohli spokojit, ale přesto toto ověřme, například pomocí grafu zachycujícího prvních sto členů, tedy sto hodnot určitého integrálu.

```
> barplot:=proc(values::list) local i, line;
> line:=(n,x)->([n-0.5,x],[n+0.5,x]);
> plot([seq(line(i,values[i]), i=1..nops(values))], args[2..-1]);
> end:
> barplot([seq(evalf(int(fn(i,x),x=-5..5)),i=1..50)], color=blue);
```



Obr. 1.8: Graf zachycující prvních padesát členů posloupnosti $\left\{\int_{-5}^{5} f_n(x)\,\mathrm{d}x\right\}_{n=1}^{\infty}$.

Výsledný graf je na obrázku 1.8.

Nyní přistupme k výpočtům. Hledejme limitu číselné posloupnosti:

```
> limit(int(fn(n,x),x=-5..5),n=infinity) assuming n::posint;
```

$$0$$

Podle diskutované věty je rovna nule i limita posloupnosti $\left\{\int_{a}^{b} f_n(x)\,\mathrm{d}x\right\}_{n=1}^{\infty}$, kde $a, b \in \mathbb{R}$.

```
> limit(int(fn(n,x),x=a..b),n=infinity) assuming n::posint, a::realcons,
  b::realcons;
```

$$0$$

Na závěr ještě určeme limitu posloupnosti $\left\{\int_{-\infty}^{\infty} f_n(x)\,\mathrm{d}x\right\}_{n=1}^{\infty}$.

```
> limit(int(fn(n,x),x=-infinity..infinity),n=infinity) assuming n::posint;
```

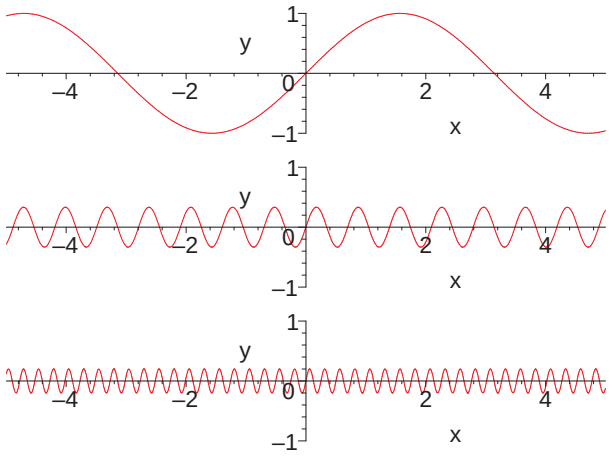
$$1$$

Výsledek byl očekávatelný, všechny členy posloupnosti jsou totiž rovny 1.

```
> int(fn(n,x),x=-infinity..infinity) assuming n::posint;
```

$$1$$

Tento příklad ukazuje, že požadavek na omezenost intervalu je nezbytný a při integrování přes neomezený interval, zde interval $(-\infty, \infty)$, již nelze pořadí limity a integrálu zaměnit. □



Obr. 1.9: Vybrané náhledy z animace, grafy funkcí f_1, f_3, f_5.

Derivování posloupnosti člen po členu

Po zkušenostech s určitým integrálem vyvstává otázka, za jakých podmínek můžeme zaměnit pořadí derivace a limity, tj.

$$\left(\lim_{n\to\infty} f_n(x)\right)' = \lim_{n\to\infty} f_n'(x),$$

Jak ukazuje následující příklad, stejnoměrná konvergence posloupnosti $\{f_n(x)\}$ není postačující podmínkou.

Příklad 1.17. *Mějme posloupnost* $\{f_n(x)\}_{n=1}^{\infty}$ *zadanou předpisem*

$$f_n(x) = \frac{\sin(n^2 x)}{n},$$

kde $x \in \mathbb{R}$. *Vyšetřete stejnoměrnou konvergenci této posloupnosti a dále také konvergenci posloupnosti* $\{f_n'(x)\}_{n=1}^{\infty}$.

Řešení.

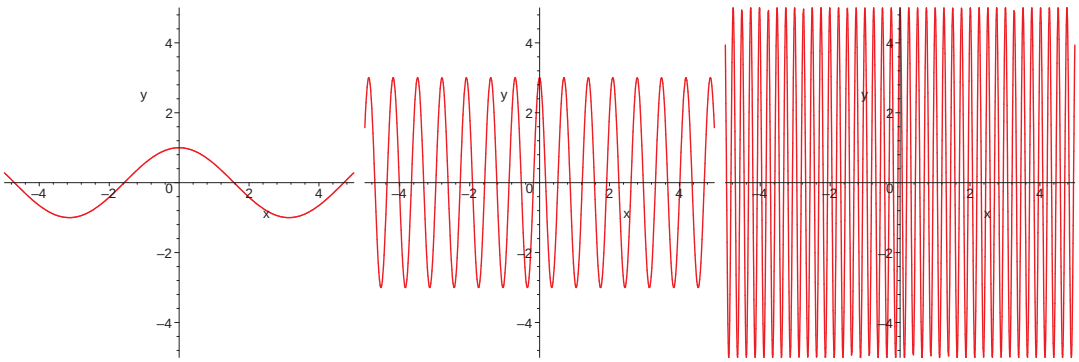
```
> fn:=(n,x)->sin(n^2*x)/n:
> plots[animate](fn(i,x),x=-5..5, i=1..10,frames=10,scaling=constrained,
   numpoints=300);
```

Vybrané náhledy z animace jsou na obrázku 1.9. Z animace je patrné, že posloupnost konverguje stejnoměrně k funkci $y = 0$. Nyní se zaměřme na posloupnost prvních derivací.

```
> diff(fn(n,x),x);
```

$$\cos(n^2 x)n$$

```
> dfn:=(n,x)->cos(n^2*x)*n:
> plots[animate](dfn(i,x),x=-5..5,i=1..10,frames=10,scaling=constrained,
   numpoints=300);
```



Obr. 1.10: Vybrané náhledy z animace, grafy derivací funkcí f_1, f_3, f_5.

V tomto případě nelze o konvergenci vůbec mluvit. V bodě $x = 0$ má posloupnost nevlastní limitu a v ostatních bodech limita neexistuje vůbec. Vybrané náhledy z animace jsou na obrázku 1.10. □

Předchozí příklad demonstroval, že derivace stejnoměrně konvergentní posloupnosti konvergovat nemusí. Postačující podmínku nám dává až následující věta:

Věta. Buď $\{f_n(x)\}$ posloupnost funkcí, které mají na otevřeném intervalu I derivaci. Nechť $\{f_n(x)\}$ konverguje na I a $\{f'_n(x)\}$ konverguje stejnoměrně na I. Pak funkce $f(x) = \lim f_n(x)$ má na I derivaci a platí $f'(x) = \lim f'_n(x)$, tj.

$$\Big(\lim_{n\to\infty} f_n(x)\Big)' = \lim_{n\to\infty} f'_n(x).$$

Kapitola 2

Řady funkcí

V této kapitole se budeme věnovat nekonečným řadám funkcí, zejména jejich konvergenci a výpočtu oboru konvergence pomocí limitního podílového a limitního odmocninového kritéria. Obor konvergence je dalším pojmem, který lze názorně prezentovat pomocí animace. Krátce se proto budeme věnovat také tomuto tématu. Na závěr se zmíníme o použití dalších kritérií konvergence pro funkční řady v programu Maple.

Uveďme pro úplnost definici nekonečné řady funkcí spolu s definicí bodové konvergence řad funkcí.

Definice. Nechť $\{f_n(x)\}_{n=1}^{\infty}$ je posloupnost funkcí definovaných na intervalu I. Symbol

$$\sum_{n=1}^{\infty} f_n(x) \quad \text{nebo } f_1(x) + f_2(x) + \cdots + f_n(x) + \cdots \tag{2.1}$$

nazýváme *nekonečnou řadou funkcí*. Posloupnost $\{s_n(x)\}_{n=1}^{\infty}$, kde $s_n(x) = f_1(x) + \cdots + f_n(x)$, nazýváme *posloupností částečných součtů* řady $\sum_{n=1}^{\infty} f_n(x)$.

Jestliže posloupnost částečných součtů $\{s_n(x)\}_{n=1}^{\infty}$ konverguje pro všechna $x \in I$, řekneme, že řada (2.1) *bodově konverguje* na intervalu I a funkci $s(x) = \lim s_n(x)$ nazýváme *součtem řady* $\sum f_n(x)$.

2.1 Reprezentace řad funkcí

Při výpočtech často nepracujeme s řadou samotnou, ale s posloupností členů funkční řady. Řadu pak můžeme reprezentovat posloupností částečných součtů, tedy některou z již dříve zmíněných možností. S výhodou využijeme procedury `sum`. Stejnou proceduru můžeme využít pro výpočet součtu funkční řady, proceduru `Sum` naopak k jejímu formálnímu zápisu.

Uvažme funkční řadu $\sum_{n=0}^{\infty} \ln^n(x)$. Již známým způsobem reprezentujeme posloupnost jejích členů. Posloupnost částečných součtů definujeme buď pomocí posloupnosti členů funkční řady:

```
> fn:=(n,x)->ln(x)^n;
```

$$fn := (n, x) \to \ln(x)^n$$

```
> sn:=(n,x)->sum(fn(k,x), k=0..n);
```

$$sn := (n, x) \to \sum_{k=0}^{n} fn(k, x)$$

A nebo přímo:

```
> sn:=(n,x)->sum(ln(x)^k, k=0..n);
```

$$sn := (n, x) \to \sum_{k=0}^{n} \ln(x)^k$$

K jednotlivým částečným součtům pak přistupujeme známým způsobem.

```
> sn(3,x);
```

$$1 + \ln(x) + \ln(x)^2 + \ln(x)^3$$

Součet řady pak určíme voláním

```
> sn(infinity,x);
```

$$-\frac{1}{\ln(x) - 1}$$

nebo explicitně příkazem

```
> sum(ln(x)^k, k=0..infinity);
```

Často však Maple nedovede součet určit. V takovém případě je výsledkem procedury formální zápis funkční řady. Přímo tento formální zápis získáme pomocí procedury `Sum`.

```
> Sum(ln(x)^n, n=0..infinity);
```

$$\sum_{n=0}^{\infty} \ln(x)^n$$

Takto zapsanou funkční řadu opět můžeme sečíst, je-li toho Maple schopen, pomocí procedury `value`.

```
> value(%);
```

$$-\frac{1}{\ln(x) - 1}$$

2.2 Výpočet oboru konvergence

Podobně jako v případě posloupností funkcí, největší množinu bodů (vzhledem k množinové inkluzi), na níž daná nekonečná řada funkcí konverguje, nazýváme *oborem konvergence* řady funkcí $\sum f_n(x)$.

V příkladech z předchozí kapitoly jsme hledali limity posloupností funkcí, ale obor konvergence byl již specifikován v zadání. V této části si na řešení několika příkladů předvedeme použití programu Maple při určování oboru konvergence funkčních řad. Opět budeme nejdříve postupovat „krok za krokem“, podobně jako při ručním výpočtu, a poté použijeme proceduru, která tento postup automatizuje.

Při výpočtech budeme využívat limitního podílového a limitního odmocninového kritéria konvergence pro číselné řady a to v následující podobě.

Věta (Limitní podílové kritérium)**.** Buď $\sum a_n$ číselná řada s nenulovými členy. Existuje-li limita $\lim \left|\frac{a_{n+1}}{a_n}\right| = q$, kde $q \in \mathbb{R}^*$, pak v případě $q < 1$ řada $\sum a_n$ konverguje absolutně a v případě $q > 1$ tato řada diverguje.

Věta (Limitní odmocninové kritérium)**.** Buď $\sum a_n$ číselná řada. Existuje-li limita $\sqrt[n]{|a_n|} = q$, kde $q \in \mathbb{R}^*$, pak v případě $q < 1$ řada $\sum a_n$ konverguje absolutně a v případě $q > 1$ tato řada diverguje.

Příklad 2.1. *Určete obor konvergence funkční řady*

$$\sum_{n=0}^{\infty} \mathrm{e}^{-n^2 x}.$$

Řešení. Definujeme funkci reprezentující jednotlivé členy řady.

```
> an:=(n,x)->exp(-n^2*x):
```

Při řešení úlohy využijeme limitního odmocninového kritéria. Nejdříve tedy určíme limitu

$$L = \lim_{n\to\infty} \sqrt[n]{|a_n|}.$$

Spočítáme n-tou odmocninu a výraz zjednodušíme.

```
> assume(x::realcons,n::posint);
> abs(an(n,x))^(1/n);
```

$$\mathrm{e}^{(-n^2 x)\left(\frac{1}{n}\right)}$$

```
> simplify(%);
```

$$\mathrm{e}^{(-nx)}$$

Následně určíme limitu pro $n \to \infty$.

```
> L:=limit(%,n=infinity);
```

$$L := \lim_{n\to\infty} \mathrm{e}^{(-nx)}$$

Limitu se nepodařilo určit. Půjdeme proto s limitou do exponentu. Je důležité si uvědomit, proč si to můžeme právě v tomto případě dovolit.

```
> L:=exp(limit(-n*x,n=infinity));
```

$$L := \mathrm{e}^{(-signum(x)\infty)}$$

Vyřešíme, pro která x je splněna podmínka konvergence z limitního odmocninového kritéria, tedy pro která x je $L < 1$.

```
> solve(L<1,x);
```

$$RealRange(Open(0), \infty)$$

Tedy pro $x \in (0, \infty)$ je nekonečná řada konvergentní. Zbývá ještě vyřešit otázku konvergence v krajním bodě $x = 0$. Divergence řady v tomto bodě je patrná na první pohled. Z důvodu ilustrace postupu výpočet přesto provedeme.

Využijeme proceduru `csum` z Maple Advisor Database, která slouží k ověřování konvergence číselných řad. Procedura implementuje několik kritérií konvergence pro číselné řady a proto je k tomuto účelu vhodnější, než standardní procedura `sum`. Bližší informace o průběhu výpočtu získáme pomocí příkazu `infolevel[csum]:=2:`.

```
> infolevel[csum]:=2:
> csum(subs(x=0, an(n)), n);
  csum: Checking sum of exp(0) over n
  csum/limitzero: Checking limit of terms
  csum/limitzero: Diverges, limit of terms is exp(0)
  csum: Series diverges
```

$$false$$

V bodě $x = 0$ tedy řada diverguje. Oborem konvergence zkoumané funkční řady je tedy interval $(0, \infty)$. □

Příklad 2.2. *Určete obor konvergence funkční řady*

$$\sum_{n=0}^{\infty} x^n \mathrm{tg}\left(\frac{x}{2^n}\right).$$

Řešení.

```
> an:=(n,x)->x^n*tan(x/2^n):
```

Při řešení užijeme limitního podílového kritéria. Nejdříve tedy určíme limitu

$$L = \lim_{n\to\infty} \left|\frac{a_{n+1}}{a_n}\right|.$$

```
> simplify(abs(an(n+1,x)/an(n,x)));
```

$$\frac{\sin\left(x2^{(-1-n)}\right)\cos\left(x2^{(-n)}\right)|x|}{\cos\left(x2^{(-1-n)}\right)\sin\left(x2^{(-n)}\right)}$$

```
> L:=limit(%, n=infinity);
```

$$L := \frac{1}{2}|x|$$

Vyřešíme, pro která x je splněna podmínka konvergence z limitního odmocninového kritéria, tedy pro která x je $L < 1$.

```
> solve(L<1,x);
```

$$RealRange(Open(-2), Open(2))$$

Limitním odmocninovým kritériem nelze o konvergenci řady rozhodnout v krajních bodech $x = -2$ a $x = 2$, v nichž je limita $L = 1$. Konvergenci ověříme opět pomocí procedury `csum`.

```
> subs(x=-2, an(n,x));
```

$$(-2)^n \tan\left(-\frac{2}{2^n}\right)$$

```
> csum(%, n); csum: Checking sum of -(-2)^n*tan(2/(2^n)) over n
  csum: Alternate terms -4^k*tan(2*4^(-k)) and 2*4^k*tan(4^(-k))
  csum/limitzero: Checking limit of terms
  csum/limitzero: Diverges, limit of terms is -2
  csum/alternate: Series diverges
```

$$false$$

```
> subs(x=2, an(n,x));
```

$$2^n \tan\left(\frac{2}{2^n}\right)$$

```
> simplify(%);
```

$$2^n \tan\left(2^{(1-n)}\right)$$

```
> csum(%, n); csum: Checking sum of 2^n*tan(2^(1-n)) over n
  csum/limitzero: Checking limit of terms
  csum/limitzero: Diverges, limit of terms is 2
  csum: Series diverges
```

$$false$$

V krajních bodech intervalu zkoumaná funkční řada diverguje, její obor konvergence tak tvoří interval $(-2, 2)$. □

Procedura `TestSeriesConvergence`

V tomto odstavci si představíme proceduru `TestSeriesConvergence`, která automatizuje postup použitý v předcházejících příkladech.

Procedura očekává tři parametry. Prvním je algebraický výraz popisující členy funkční řady. Druhým je proměnná reprezentující index a třetím neznámá uvedených funkcí. Použití procedury si předvedeme na několika příkladech, přičemž opět necháme zobrazovat bližší informace o výpočtu pomocí příkazu `infolevel[TestSeriesConvergence]:=2:`. Příkaz samotný již v řešení uvádět nebudeme.

Příklad 2.3. *Určete obor konvergence funkční řady*

$$\sum_{n=0}^{\infty} \frac{n}{(n+1)(3x^2+4x+2)^n}.$$

Řešení.

```
> an:=(n,x)->n/((n+1)*(3*x^2+4*x+2)^n):
> TestSeriesConvergence(an(n,x),n,x);
  TestSeriesConvergence: Trying ration test
  TestSeriesConvergence: Limit: 1/abs(3*x^2+4*x+2)
  TestSeriesConvergence: Range solved: RealRange(-infinity,Open(-1)),
  RealRange(Open(-1/3),infinity)
  TestSeriesConvergence: Checking isolated points
  TestSeriesConvergence: Diverges at point x = -1
  TestSeriesConvergence: Diverges at point x = -1/3
```

$$RealRange\left(-\infty, Open(-1)\right), RealRange\left(Open\left(-\frac{1}{3}\right), \infty\right)$$

Z výpisu je patrné, že pro výpočet bylo zvoleno limitní podílové kritérium a vypočítána limita

$$L = \frac{1}{|3x^2 + 4x + 2|}.$$

Dále byl určen interval, na kterém je splněna podmínka konvergence $L < 1$ a konečně také (pomocí procedury `csum`) ověřena konvergence v bodech, u nichž nelze tímto kritériem rozhodnout (zde krajní body intervalů $x = -1$ a $x = -\frac{1}{3}$).

Oborem konvergence zkoumané řady je sjednocení intervalů $(-\infty, -1) \cup \left(-\frac{1}{3}, \infty\right)$. □

Příklad 2.4. *Určete obor konvergence funkční řady*

$$\sum_{n=0}^{\infty} \ln^n(x).$$

Řešení.

```
> an:=(n,x)->ln(x)^n:
> TestSeriesConvergence(an(n,x),n,x);
  TestSeriesConvergence: Series identical zero when x = 1
  TestSeriesConvergence: Trying ration test
  TestSeriesConvergence: Limit: abs(ln(x))
  TestSeriesConvergence: Range solved: RealRange(Open(1/exp(1)),Open(exp(1)))
  TestSeriesConvergence: Checking isolated points
  TestSeriesConvergence: Test failed at point x = 1/exp(1)
  TestSeriesConvergence: Diverges at point x = exp(1)
```

$$RealRange\left(Open\left(\frac{1}{\mathrm{e}}\right), Open(\mathrm{e})\right)$$

Z výpisu je patrné, že v bodě $x = \frac{1}{\mathrm{e}}$ test selhal, proto tento případ blíže prozkoumáme. Dosazením bodu $x = \frac{1}{\mathrm{e}}$ získáme alternující číselnou řadu $\sum_{n=0}^{\infty}(-1)^n$. Jedná se o tzv. *Grandiho řadu*, která je divergentní (posloupnost částečných součtů této řady nemá limitu).

Oborem konvergence zkoumané řady je interval $\left(\frac{1}{\mathrm{e}}, \mathrm{e}\right)$. □

Příklad 2.5. *Určete obor konvergence funkční řady*

$$\sum_{n=1}^{\infty} \frac{(-1)^{n-1}x^{2n}}{\sqrt{n}(2n-1)}.$$

Řešení.

```
> an:=(n,x)->(-1)^(n-1)*x^(2*n)/sqrt(n)/(2*n-1):
> TestSeriesConvergence(an(n,x),n,x);
  TestSeriesConvergence: Series identical zero when x = 0
  TestSeriesConvergence: Trying ration test
  TestSeriesConvergence: Limit: x^2
  TestSeriesConvergence: Range solved: RealRange(Open(-1),Open(1))
  TestSeriesConvergence: Checking isolated points
  TestSeriesConvergence: Converges at point x = 1
  TestSeriesConvergence: Converges at point x = -1
```

$$RealRange(-1, 1)$$

V tomto případě vyšetřovaná řada konverguje i v krajních bodech intervalu, jejím oborem konvergence je tedy interval $[-1, 1]$. □

2.3 Další metody ověřování konvergence funkčních řad

Dříve představená procedura `TestSeriesConvergence` automatizuje výpočet oboru konvergence funkční řady, přičemž využívá limitního podílového a limitního odmocninového kritéria. K ověření konvergence funkční řady lze využít i jiná kritéria. Jistou jejich nevýhodou je, že je velmi obtížné tyto testy automatizovat. Program Maple tak lze využít pouze k mezivýpočtům, postup vedoucí k řešení úlohy určujeme sami.

Často využívané kritérium je tzv. *Weierstrassovo kritérium.*

Věta. Nechť $\{f_n(x)\}$ je posloupnost funkcí na I. Nechť existuje posloupnost nezáporných čísel $\{a_n\}$ taková, že řada $\sum a_n$ konverguje a platí

$$|f_n(x)| \leq a_n \quad \text{pro všechna } x \in I \text{ a } n \in \mathbb{N}.$$

Pak řada $\sum f_n(x)$ konverguje stejnoměrně na intervalu I.

Aplikaci tohoto kritéria a zároveň i úskalí vzniklá při výpočtu prováděném v programu Maple si předvedeme v následujícím příkladu.

Příklad 2.6. *Rozhodněte o konvergenci funkční řady*

$$\sum_{n=1}^{\infty} \frac{\sin(nx)\cos(nx)}{n^2}.$$

Řešení. Víme, že řada $\sum_{n=1}^{\infty} \frac{K}{n^2}$ je pro pevně zvolené K absolutně konvergentní. Dokážeme-li, že funkce $\sin(nx)\cos(nx)$ je ohraničená, tj. že existuje $K \in \mathbb{R}^+$ takové, že pro všechna $x \in \mathbb{R}$ a $n \in \mathbb{N}$ platí

$$-K < \sin(nx)\cos(nx) < K,$$

dokážeme i stejnoměrnou (a absolutní) konvergenci řešené řady.

Program Maple bohužel nenabízí spolehlivý nástroj, jak zjistit, zda je konkrétní funkce ohraničená. První možností jsou funkce `maximize` a `minimize`.

```
> expr:=sin(n*x)*cos(n*x):
> maximize(expr,x=0..2*Pi,n=1..infinity);[1]
  Error, (in maximize) sort: 2nd argument must be a function that always returns true
  or false
```

Zde výpočet zhavaroval z důvodu chyby v proceduře `maximize`.[2]

Druhou možností je pomocí procedury `evalr` vyhodnotit výraz s využitím intervalové aritmetiky.

```
> evalr(subs(x=INTERVAL(0..2*Pi),expr));
```

$$INTERVAL(-1..1)$$

Získaný výsledek nemusí být na první pohled podezřelý. Zkusme ale výraz upravit.

```
> convert(expr,sin);
```

$$\frac{1}{2}\sin(2nx)$$

```
> evalr(subs(x=INTERVAL(0..2*Pi),%));
```

$$INTERVAL\left(-\frac{1}{2}..\frac{1}{2}\right)$$

Získaný výsledek, tentokrát správný, se od předchozího liší. Oba výsledky sice ukazují, že funkce $\sin(nx)\cos(nx)$ je na $\mathbb{R}$ ohraničená a tedy zkoumaná řada konverguje absolutně a stejnoměrně, ale rozdílné výsledky by nás měly varovat.

Tímto poměrně snadným příkladem jsme chtěli demonstrovat, že procedury `maximize` a `evalr` nejsou pro naše účely příliš vhodné. Nám tak opět nezbývá nic jiného, než se při řešení spolehnout na vlastní znalosti a úsudek. □

2.4 Ilustrování konvergence funkčních řad animací

Při vytváření animací je vhodné obor konvergence řady barevně vyznačit, například pomocí barevného pásu. K tomu využijeme proceduru `plotVerticalStrip`.

[1] Funkce $\sin nx$ a $\cos nx$ jsou 2π-periodické, proto se stačí omezit například na interval $[0, 2\pi]$.

[2] Chyba není způsobena nesprávným použitím funkce `maximize`, v případě méně komplikované varianty `maximize(sin(n*x),x=0..2*Pi,n=1..infinity);` vrací Maple správný výsledek. Problém pravděpodobně způsobuje nekonečný rozsah hodnot proměnné `n`, při kterém Maple dokáže řešit úlohu pouze v triviálních případech.

Procedura očekává dva parametry. První určuje šířku pásu a jeho tvar odpovída výstupu procedury `TestSeriesConvergence`.[3] Druhý parametr určuje vertikální rozsah grafu. Výstupem procedury je graf barevného pásu, jehož okraje jsou ohraničeny plnou nebo přerušovanou čarou, v závislosti na otevřenosti resp. uzavřenosti intervalu. Barvu výplně a hraniční čáry lze nastavit pomocí parametrů `color` a `bordercolor`.

Použití procedury a vytváření animace si budeme ilustrovat na následujícím příkladě.

Příklad 2.7. *Ilustrujte pomocí animace konvergenci funkční řady*

$$\sum_{n=1}^{\infty} \frac{(-1)^n (x+2)^n}{n+\sqrt{n}}$$

uvnitř a vně oboru konvergence.

Řešení. Rozdíl v chování konvergence řady uvnitř a vně jejího oboru konvergence je více patrný na grafech posloupnosti členů řady než na posloupnosti částečných součtů. Důvodem je, že součtem řady nemusí být funkce ohraničená (na oboru konvergence) zatímco členy konvergentní funkční řady se na oboru konvergence limitně blíží nule. Sestrojíme proto animace dvě, jednu znázorňující grafy jednotlivých členů funkční řady a druhou znázorňující posloupnost částečných součtů.

V druhé animaci nebudeme vyznačovat součet funkční řady. Ikdyž se nám pro některé funkční řady podaří jejich součet určit, často jej nemá smysl v animaci vyznačovat[4] z důvodu příliš pomalé konvergence.

```
> fn:=(n,x)->(-1)^n*(x+2)^n/(n+sqrt(n)):
> TestSeriesConvergence(fn(n,x),n,x);
```

$$RealRange(Open(-3), 1)$$

Oborem konvergence řady je interval $(-3, 1]$. Nyní si připravíme graf barevného pásu a sestojíme animaci.

```
> pruh:=plotVerticalStrip(RealRange(Open(-3),-1),-2..2):
> graf:=n->plots[display](pruh, plot(fn(n,x),x=-4..0,y=-2..2,
   thickness=2)):
> plots[display](seq(graf(i),i=1..15), insequence=true,
   scaling=constrained);
```

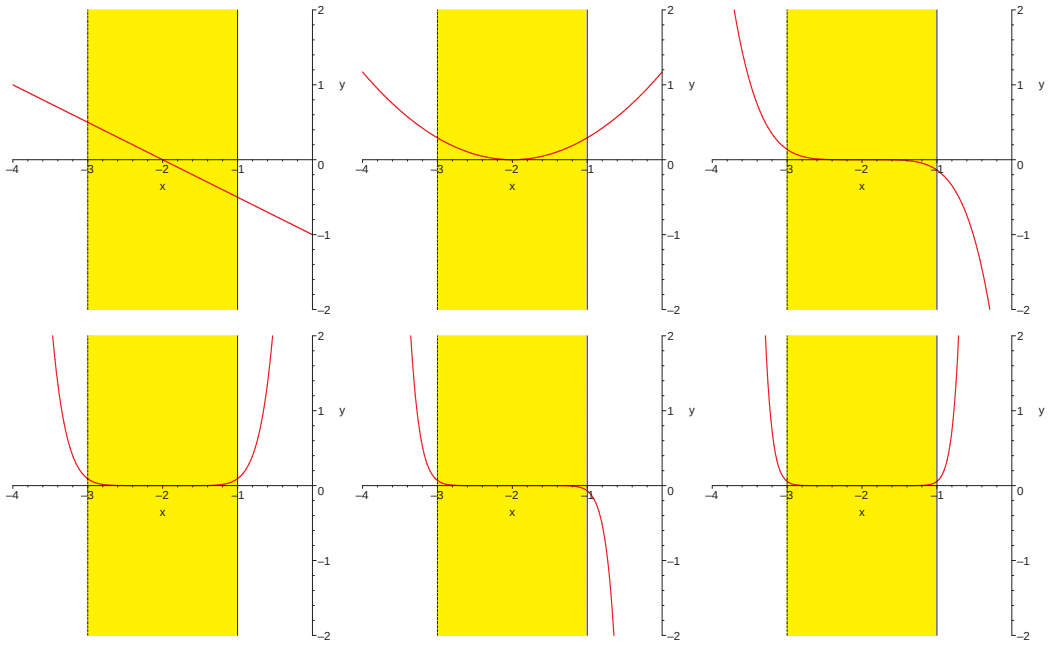
Vybrané náhledy z animace jsou na obrázku 2.1. Je na nich patrné, jak se jednotlivé funkce na oboru konvergence blíží nule, naopak vně oboru konvergene funkce velmi rychle rostou nebo klesají.

Podobně postupujeme při vytváření animace s grafy částečných součtů.

```
> sn:=(n,x)->sum(fn(k,x),k=1..n):
> pruh:=plotVerticalStrip(RealRange(Open(-3),-1),-1.5..4):
```

[3] Tedy například `RealRange(Open(-1),1)`.

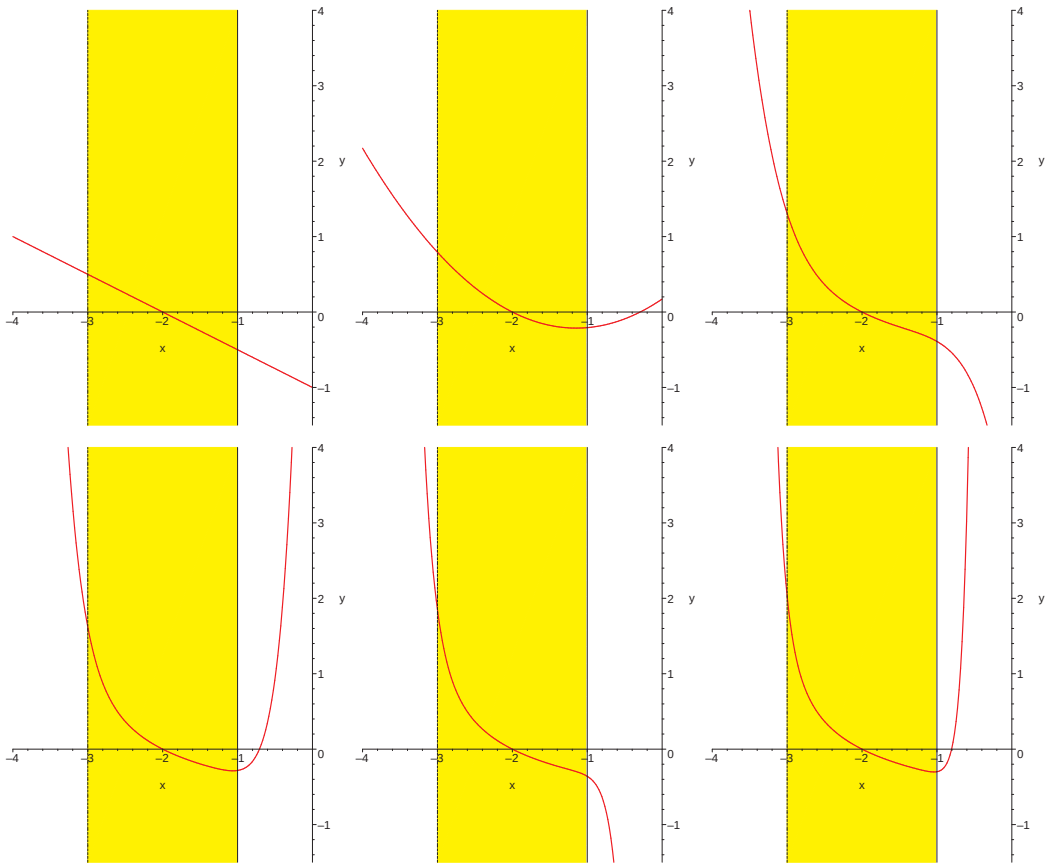
[4] Jako jsme například vyznačovali limitu při ilustrování stejnoměrně konvergence posloupností funkcí.



Obr. 2.1: Vybrané náhledy z animace (grafy funkcí f_1, f_2, f_5, f_8, f_{11} a f_{14}).

```
> graf:=n->plots[display](pruh, plot(sn(n,x),x=-4..0,y=-1.5..4,
   thickness=2)):
> plots[display](seq(graf(i),i=1..15), insequence=true,
   scaling=constrained);
```

Náhledy z animace jsou na obrázku 2.2. □



Obr. 2.2: Vybrané náhledy z animace (grafy částečných součtů s_1, s_2, s_5, s_8, s_{11} a s_{14}).

Kapitola 3

Mocninné řady

V této kapitole se budeme věnovat speciálnímu typu nekonečných řad funkcí a to řadám mocninným.

Definice. Buď $\{a_n\}_{n=0}^{\infty}$ posloupnost reálných čísel, x_0 libovolné reálné číslo. *Mocninnou řadou* se středem v bodě x_0 a koeficienty a_n rozumíme řadu funkcí tvaru

$$a_0 + a_1(x - x_0) + a_2(x - x_0)^2 + \cdots + a_n(x - x_0)^n + \cdots = \sum_{n=0}^{\infty} a_n(x - x_0)^n.$$

Poznámka. Pro $x_0 \neq 0$ lze pomocí substituce $x - x_0$ převést mocninnou řadu se středem v bodě x_0 na mocninnou řadu se středem v bodě 0. Bez újmy na obecnosti lze tedy předpokládat, že středem mocninné řady je číslo $x_0 = 0$.

3.1 Obor konvergence mocninné řady

V předchozí kapitole jsme ilustrovali výpočet oboru konvergence funkční řady pomocí limitního podílového respektive limitního odmocninového kritéria. Stejný postup lze samozřejmě použít také v případě řad mocninných. Díky jejich speciálnímu tvaru lze však tento výpočet mírně zjednodušit. Přesněji obor konvergence mocninných řad popisuje následující věta.

Věta. Nechť $\sum a_n x^n$ je mocninná řada a nechť

$$a = \limsup \sqrt[n]{|a_n|}.$$

Je-li $a = 0$, pak řada absolutně konverguje pro všechna $x \in \mathbb{R}$, říkáme, že řada vždy konverguje.

Je-li $a = \infty$, pak řada diverguje pro všechna $x \neq 0$, říkáme, že řada vždy diverguje.

Je-li $0 < a < \infty$, pak řada absolutně konverguje pro $|x| < \frac{1}{a}$ a diverguje pro $|x| > \frac{1}{a}$.

Je-li $0 < a < \infty$, pak se číslo $r = \frac{1}{a}$ nazývá *poloměr konvergence* a interval $(-r, r)$ se nazývá *konvergenční interval*. Chování řady v krajních bodech konvergenčního intervalu je třeba vyšetřit zvlášť, protože závisí na tvaru mocninné řady. Oborem konvergence

mocninné řady, která vždy nediverguje, je proto konvergenční interval s případnými jeho krajními body, pokud v nich řada konverguje.

Jestliže řada $\sum a_n x^n$ vždy konverguje, tj. $a = 0$, definujeme její poloměr konvergence jako $r = \infty$ a její konvergenční interval jako $(-\infty, \infty)$.

Jestliže řada $\sum a_n x^n$ vždy diverguje, tj. $a = \infty$, definujeme její poloměr konvergence jako $r = 0$.

Pro naše výpočty je důležitá následující poznámka.

Poznámka. Existuje-li $\lim \sqrt[n]{|a_n|} = a$, pak má mocninná řada $\sum a_n x^n$ poloměr konvergence

$$r = \frac{1}{\lim_{n\to\infty} \sqrt[n]{|a_n|}}$$

(přitom klademe $r = \infty$, je-li $a = 0$, a $r = 0$, je-li $a = \infty$).

Existuje-li $\lim \left|\frac{a_n}{a_{n+1}}\right|$, lze poloměr konvergence určit jako

$$r = \lim_{n\to\infty} \left|\frac{a_n}{a_{n+1}}\right|.$$

Předchozí poznámka poskytuje návod na výpočet poloměru konvergence. Postup budeme ilustrovat v následujícím příkladě.

Příklad 3.1. *Určete obor konvergence mocninné řady*

$$\sum_{n=1}^{\infty} \frac{(-1)^n (x+2)^n}{n+\sqrt{n}}.$$

Řešení. Tato řada má střed v bodě $x_0 = -2$. Určíme poloměr konvergence. Abychom výpočet usnadnili, nejdříve spočítáme a upravíme podíl $\frac{a_n}{a_{n+1}}$. Teprve poté určíme limitu

$$\lim_{n\to\infty} \left|\frac{a_n}{a_{n+1}}\right|.$$

```
> an:=n->(-1)^n/(n+sqrt(n)):
> simplify(an(n)/an(n+1));
```

$$-\frac{n+1+\sqrt{n+1}}{n+\sqrt{n}}$$

```
> r:=limit(abs(%), n=infinity);
```

$$r := 1$$

Konvergenční interval je tedy $(-3, -1)$. Zbývá vyšetřit konvergenci řady v krajních bodech tohoto intervalu. To provedeme s využitím již známé procedury `csum` z Maple Advisor Database.

```
> csum(subs(x=-3, an(n)), n);
```

$$false$$

```
> csum(subs(x=-1, an(n)), n);
```

$$true$$

Oborem konvergence je tedy interval $(-3, -1]$. □

Příklad 3.2. *Určete obor konvergence mocninné řady*

$$\sum_{n=1}^{\infty} \left(1 + \frac{1}{n}\right)^{n^2} x^n.$$

Řešení. Tato mocninná řada má střed v bodě $x_0 = 0$. Tentokrát při výpočtu poloměru konvergence využijeme limitní odmocninnové kritérium. Výpočet opět rozdělíme do dvou kroků, přičemž nejdříve určíme hodnotu výrazu $\frac{1}{\sqrt[n]{|a_n|}}$.

```
> an:=n->(1+1/n)^(n^2):
> simplify(1/abs(an(n))^(1/n));
```

$$\left|\left(\frac{n+1}{n}\right)^{(n^2)}\right|^{\left(-\frac{1}{n}\right)}$$

```
> r:=limit(%, n=infinity);
```

$$r := \mathrm{e}^{(-1)}$$

Poloměr konvergence je roven $\frac{1}{\mathrm{e}}$ a konvergenční interval je tedy $\left(-\frac{1}{\mathrm{e}}, \frac{1}{\mathrm{e}}\right)$. Vyšetříme konvergenci mocninné řady v krajních bodech tohoto intervalu.

```
> csum(subs(x=-r, an(n)), n);
```

$$false$$

```
> csum(subs(x=-r, an(n)), n);
```

$$false$$

V obou krajních bodech konvergenčního intervalu mocninná řada diverguje a oborem konvergence je tedy interval $\left(-\frac{1}{\mathrm{e}}, \frac{1}{\mathrm{e}}\right)$. □

Jak jsme již zmínili, předchozí příklady by bylo možné vyřešit také obecnějším přístupem z předchozí kapitoly. Podobně lze využít i proceduru `TestSeriesConvergence`.

Příklad 3.3. *Určete obor konvergence mocninné řady*

$$\sum_{n=1}^{\infty} \frac{2^n}{n^2} x^n.$$

Řešení.

```
> an:=(n,x)->2^n*x^n/n^2:
> TestSeriesConvergence(an(n,x),n,x);
  TestSeriesConvergence: Series identical zero when x = 0
  TestSeriesConvergence: Trying ration test
  TestSeriesConvergence: Limit: 2*abs(x)
  TestSeriesConvergence: Range solved: RealRange(Open(-1/2),Open(1/2))
  TestSeriesConvergence: Checking isolated points
  TestSeriesConvergence: Converges at point x = 1/2
  TestSeriesConvergence: Converges at point x = -1/2
```

$$RealRange\left(-\frac{1}{2},\frac{1}{2}\right)$$

Vyšetřovaná řada má střed v bodě $x_0 = 0$. Oborem konvergence je interval $\left[-\frac{1}{2},\frac{1}{2}\right]$, z čehož lze určit poloměr konvergence $r = \frac{1}{2}$. V obou krajních bodech konvergenčního intervalu mocninná řada konverguje. □

Příklad 3.4. *Určete obor konvergence mocninné řady*

$$\sum_{n=1}^{\infty} \frac{(n!)^2}{(2n)!} x^n.$$

Řešení.

```
> an:=(n,x)->(n!)^2*x^n/(2*n)!:
> TestSeriesConvergence(an(n,x),n,x);
  TestSeriesConvergence: Series identical zero when x = 0
  TestSeriesConvergence: Trying ration test
  TestSeriesConvergence: Limit: 1/4*abs(x)
  TestSeriesConvergence: Range solved: RealRange(Open(-4),Open(4))
  TestSeriesConvergence: Checking isolated points
  TestSeriesConvergence: Diverges at point x = 4
  TestSeriesConvergence: Diverges at point x = -4
```

$$RealRange\left(Open(-4), Open(4)\right)$$

Vyšetřovaná řada má střed v bodě $x_0 = 0$. Oborem konvergence je interval $(-4, 4)$, poloměr konvergence $r = 4$. V obou krajních bodech konvergenčního intervalu mocninná řada diverguje. □

3.2 Součet a vlastnosti mocninných řad

Pro nalezení součtu mocninné řady lze využít proceduru `sum`. Tato procedura používá k nalezení součtu zadané řady několik složitých algoritmů, bližší informace o průběhu výpočtu získáme opět pomocí proměnné prostředí `infolevel`, tj. příkazem `infolevel[sum]:=2:`

Příklad 3.5. *Určete obor konvergence a součet mocninné řady*

$$\sum_{n=0}^{\infty} 2^n x^{2n}.$$

Řešení.

```
> an:=(n,x)->2^n*x^(2*n):
> TestSeriesConvergence(an(n,x),n,x);
  TestSeriesConvergence: Series identical zero when x = 0
  TestSeriesConvergence: Trying ration test
  TestSeriesConvergence: Limit: 2*x^2
  TestSeriesConvergence: Range solved: RealRange(Open(-1/2*2^(1/2)),Open(1/2*2^(1/2)))
  TestSeriesConvergence: Checking isolated points
  TestSeriesConvergence: Diverges at point x = 1/2*2^(1/2)
  TestSeriesConvergence: Diverges at point x = -1/2*2^(1/2)
```

$$RealRange\left(Open\left(-\frac{\sqrt{2}}{2}\right), Open\left(\frac{\sqrt{2}}{2}\right)\right)$$

```
> sum(an,n=0..infinity);
  sum/infinite: infinite summation
  sum/indefnew: indefinite summation
  sum/indefnew: indefinite summation finished
  sum/def2: definite summation using hypergeometric fcns
  convert/hypergeom/from[sum] Function 2^n*x^(2*n) satisfies the criteria
```

$$-\frac{1}{2x^2-1}$$

Vyšetřovaná řada má střed v bodě $x_0 = 0$. Oborem konvergence je interval $\left(-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$, z čehož lze určit poloměr konvergence $r = \frac{\sqrt{2}}{2}$. V krajních bodech konvergenčního intervalu mocninná řada diverguje. Součtem mocninné řady je funkce $y = -\frac{1}{2x^2-1}$. □

Příklad 3.6. *Určete obor konvergence a součet mocninné řady*

$$\sum_{n=1}^{\infty} (-1)^{n+1} \frac{x^{n+1}}{n(n+1)}.$$

Řešení.

```
> an:=(n,x)->(-1)^(n+1)*(x^(n+1))/(n*(n+1)):
> TestSeriesConvergence(an(n,x),n,x);
  TestSeriesConvergence: Series identical zero when x = 0
  TestSeriesConvergence: Trying ration test
```

```
TestSeriesConvergence: Limit: abs(x)
TestSeriesConvergence: Range solved: RealRange(Open(-1),Open(1))
TestSeriesConvergence: Checking isolated points
TestSeriesConvergence: Converges at point x = 1
TestSeriesConvergence: Converges at point x = -1
```

$$RealRange(-1, 1)$$

```
> sum(an(n,x),n=1..infinity);
convert/hypergeom/from: Function -(-1)^(x+1)*x^(x+2)/((x+1)*(x+2)) satisfies the criteria
```

$$\ln(x+1)x + \ln(x+1) - x$$

Mocninná řada má střed v bodě $x_0 = 0$, poloměr konvergence $r = 1$. Oborem konvergence zkoumané řady je interval $[-1, 1]$, součtem této řady je funkce $y = \ln(x+1)x + \ln(x+1) - x$. □

Pokud se nepodaří proceduře `sum` nalézt součet mocninné řady a nebo nejsme s obdrženou odpovědí spokojeni, nezbývá, než se vrátit k ručnímu výpočtu. Přitom lze s výhodou využít vlastností mocninných řad, z nichž některé jsou shrnuty v následujících tvrzeních.

Věta. Nechť $r > 0$ je poloměr konvergence mocninné řady $\sum a_n x^n$. Pak tato řada stejnoměrně konverguje na každém uzavřeném podintervalu $[-\rho, \rho]$ intervalu $(-r, r)$.

Důsledek. Nechť mocninná řada $\sum a_n x^n$ má poloměr konvergence $r > 0$. Pak součet této řady je funkce spojitá na intervalu $(-r, r)$.

Důsledek. Nechť mocninná řada $\sum a_n x^n$ má poloměr konvergence $r > 0$. Pak pro všechna $x \in (-r, r)$ platí

$$\int_0^x \left(\sum_{n=0}^{\infty} a_n t^n \right) \mathrm{d}t = \sum_{n=0}^{\infty} \int_0^x a_n t^n \, \mathrm{d}t = \sum_{n=0}^{\infty} a_n \frac{x^{n+1}}{n+1},$$

přičemž mocninná řada na pravé straně má stejný poloměr konvergence r.

Důsledek. Nechť mocninná řada $\sum a_n x^n$ má poloměr konvergence $r > 0$. Pak součet této řady je funkce spojitá na intervalu $(-r, r)$.

Důsledek. Nechť mocninná řada $\sum a_n x^n$ má poloměr konvergence $r > 0$. Pak pro libovolný interval $[a, b] \subset (-r, r)$ platí

$$\int_a^b \left(\sum_{n=0}^{\infty} a_n x^n \right) \mathrm{d}x = \sum_{n=0}^{\infty} \int_a^b a_n x^n \, \mathrm{d}x = \sum_{n=0}^{\infty} a_n \frac{b^{n+1}}{n+1} - \sum_{n=0}^{\infty} a_n \frac{a^{n+1}}{n+1}$$

přičemž mocninná řada na pravé straně má opět poloměr konvergence r.

Důsledek. Nechť mocninná řada $\sum a_n x^n$ má poloměr konvergence $r > 0$. Pak pro všechna $x \in (-r, r)$ platí

$$\left(\sum_{n=1}^{\infty} a_n x^n\right)' = \sum_{n=1}^{\infty} (a_n x^n)' = \sum_{n=1}^{\infty} n a_n x^{n-1},$$

přičemž mocninná řada na pravé straně má opět poloměr konvergence r.

Uvedených tvrzení využijeme v následujícím příkladu.

Příklad 3.7. *Určete poloměr konvergence a součet mocninné řady*

$$\sum_{n=1}^{\infty} \frac{x^{(4n-3)}}{4n-3} \tag{3.1}$$

Řešení.

```
> an:=(n,x)->x^(4*n-3)/(4*n-3):
> TestSeriesConvergence(an(n,x),n,x);
```

$$RealRange\,(Open(-1), Open(1))$$

Oborem konvergence mocninné řady je tedy interval $(-1, 1)$.

```
> assume(x::RealRange(Open(-1),Open(1))):
> sum(an(n,x), n=1..infinity);
```

$$\frac{1}{4} x\, LerchPhi\left(x^4, 1, \frac{1}{4}\right)$$

Vyhledáme-li v nápovědě programu Maple funkci `LerchPhi`, zjistíme, že je definovaná předpisem

$$LerchPhi(z, a, v) = \sum_{n=0}^{\infty} \frac{z^n}{(v+n)^a},$$

což nás jako řešení sotva uspokojí. Budeme proto postupovat podobně jako při ručním výpočtu.

```
> Sum(an, n=1..infinity);
```

$$\sum_{n=1}^{\infty} \frac{x^{(4n-3)}}{4n-3}$$

Řadu derivujeme člen po členu a určíme součet nově vzniklé řady.

```
> diff(%,x);
```

$$\sum_{n=1}^{\infty} \frac{x^{(4n-3)}}{x}$$

```
> simplify(%);
```

$$\sum_{n=1}^{\infty} x^{(4n-4)}$$

```
> s1:=value(%);
```

$$s1 := -\frac{1}{x^4-1}$$

Součet řady (3.1) nyní určíme jako

$$\sum_{n=1}^{\infty} \frac{x^{(4n-3)}}{4n-3} = \int_0^x -\frac{1}{t^4-1}\,\mathrm{d}t.$$

```
> s2:=subs(x=t,s1);
```

$$s2 := -\frac{1}{t^4-1}$$

```
> int(s2,t=0..x);
```

$$\int_0^x -\frac{1}{t^4-1}\,\mathrm{d}t$$

Vidíme, že Maple tento určitý integrál nevyhodnotí. Spočítáme tedy neurčitý integrál a dosadíme meze sami.

```
> r:=int(s2,t);
```

$$r := \frac{1}{2}\mathrm{arctanh}(t) + \frac{1}{2}\arctan(t)$$

Hyperbolický tangens můžeme vyjádřit pomocí přirozeného logaritmu a výraz tak upravit do následujícího tvaru:[1]

```
> r:=convert(op(1,r),ln)+op(2,r);
```

$$r := \frac{1}{4}\ln(t+1) - \frac{1}{4}\ln(1-t) + \frac{1}{2}\arctan(t)$$

Dosadíme meze určitého integrálu a určíme hodnotu rozdílu.

```
> eval(r,t=x)-eval(r,t=0);
```

$$\frac{1}{4}\ln(x+1) - \frac{1}{4}\ln(1-x) + \frac{1}{2}\arctan(x)$$

Oborem konvergence vyšetřované řady (3.1) je otevřený interval $(-1, 1)$, jejím součtem je funkce $y = \frac{1}{4}\ln(x+1) - \frac{1}{4}\ln(1-x) + \frac{1}{2}\arctan(x)$.

Ukažme si ještě jednu variantu výpočtu, kdy výraz $-\frac{1}{t^4-1}$ před integrací nejdříve rozložíme na parciální zlomky.

[1] Pomocí přirozeného logaritmu můžeme vyjádřit také funkci tangens. V tomto vyjádření však vystupují komplexní čísla, což v našem případě není příliš žádoucí.

```
> convert(s2,parfrac);
```

$$-\frac{1}{4(t-1)}+\frac{1}{4(t+1)}+\frac{1}{2(t^2+1)}$$

```
> int(%,t);
```

$$-\frac{1}{4}\ln(t-1)+\frac{1}{4}\ln(t+1)+\frac{1}{2}\arctan(t)$$

```
> r:=eval(%,t=x)-eval(%,t=0);
```

$$r := -\frac{1}{4}\ln(x-1)+\frac{1}{4}\ln(x+1)+\frac{1}{2}\arctan(t)+\frac{1}{4}\pi I$$

Na předchozím výsledku nás může zarazit, že v něm vystupuje komplexní číslo. My však víme, že součtem řady musí být funkce reálná. Zdánlivý rozpor se vyjasní, když si uvědomíme, že Maple pracuje v komplexním oboru a že ani výraz $\ln(x-1)$ nenabývá na námi vyšetřovaném intervalu $(-1,1)$ reálných hodnot. Ve výsledku se tak imaginární části těchto dvou výrazů odečtou a výsledkem je pro $x \in (-1,1)$ opravdu reálná funkce.

Nejsme-li přesto s výrazem spokojeni, můžeme jej upravit pomocí „triku“, kdy použijeme pouze jeho reálnou část.

```
> evalc(Re(r));
```

$$-\frac{1}{4}\ln(1-x)+\frac{1}{4}\ln(x+1)+\frac{1}{2}\arctan(x)$$

Naopak imaginární část výrazu je vskutku rovna nule.

```
> evalc(Im(r));
```

$$0$$

Po dosazení mezí bychom obdrželi již známou hodnotu součtu mocninné řady, funkci $y = \frac{1}{4}\ln(x+1) - \frac{1}{4}\ln(1-x) + \frac{1}{2}\arctan(x)$. □

3.3 Taylorovy a Maclaurinovy řady

V předchozí části jsme hledali součet mocninné řady. V této části se budeme zabývat úlohou opačnou, tj. hledáním rozvoje dané funkce do mocninné řady. Použití Taylorova rozvoje je snadný způsob, jak tuto úlohu řešit pro některé typy funkcí.

Věta (Taylorova věta)**.** Nechť f je funkce, která má derivace až do řádu $n+1$ v uzavřeném intervalu I, jehož krajní body jsou čísla x a x_0. Pak platí

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x-x_0) + \cdots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n + R_n(x), \tag{3.2}$$

kde $R_n(x)$ je tzv. *Taylorův zbytek*, pro který platí

$$R_n(x) = \frac{f^{(n+1)}(\vartheta)}{(n+1)!}(x-x_0)^{n+1}, \quad \text{kde } \vartheta \in I,\ \vartheta \neq x, x_0$$

Na základě Taylorovy věty je přirozené zavést následující definici.

Definice. Nechť funkce f má v bodě x_0 derivace všech řádů. Mocninnou řadu

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n \tag{3.3}$$

nazýváme *Taylorovou řadou* funkce f v bodě x_0. Je-li $x_0 = 0$, mluvíme o *Maclaurinově řadě*.

Obecně nemusí platit, že součet Taylorovy řady funkce f je roven samotné funkci f. Následující věta uvádí jednu z dostačujících podmínek, kdy tato rovnost platí.

Věta. Nechť funkce f má v nějakém bodě x_0 derivace všech řádů. Pak platí

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n$$

na intervalu I obsahujícím bod x_0 právě tehdy, když pro posloupnost $\{R_n(x)\}$ Taylorových zbytků platí $R_n(x) = 0$ pro všechna $x \in I$.

Poznámka. Dá se ukázat, že lze-li funkci f na nějakém intervalu I, jehož vnitřním bodem je x_0, rozvést do mocninné řady o středu x_0, pak je takový rozvoj pouze jediný a je současně Taylorovým rozvojem funkce f.

Pro výpočet Taylorova rozvoje máme v programu Maple k dispozici proceduru `taylor`, která je jednodušší variantou obecnější procedury `series`.[2] Výstup této procedury je v nápovědě programu Maple označován jako Taylorova řada ve zkráceném tvaru a prakticky odpovídá zápisu (3.2) z Taylorovy věty. Stupeň Taylorova polynomu v tomto rozvoji lze explicitně určit,[3] implicitní hodnotou je hodnota proměnné prostředí `Order`.

Příklad 3.8. *Určete Taylorovy polynomy stupňů* 1, 3, 5 *funkce* $y = \mathrm{e}^{\frac{x}{2}}$ *se středem v bodě* $x_0 = 1$.

Řešení.

```
> f:=exp(x/2):
> x0:=1:
> taylor(f,x=x0,2);
```

$$\mathrm{e}^{\left(\frac{1}{2}\right)} + \frac{1}{2}\mathrm{e}^{\left(\frac{1}{2}\right)}(x-1) + O((x-1)^2)$$

Pomocí procedury `convert` odřízneme zbytek.

[2] Proceduru `series` lze použít také například k výpočtům řad Laurentových, respektive částečných součtů těchto řad.

[3] V případě procedury `taylor` musíme zadat hodnotu o 1 vyšší, než je požadovaný stupeň.

```
> convert(%, polynom);
```

$$\mathrm{e}^{\left(\frac{1}{2}\right)} + \frac{1}{2}\mathrm{e}^{\left(\frac{1}{2}\right)}(x-1)$$

Stejným způsobem spočítáme Taylorovy polynomy stupňů 3 a 5.

```
> convert(taylor(f,x=x0,4), polynom);
```

$$\mathrm{e}^{\left(\frac{1}{2}\right)} + \frac{1}{2}\mathrm{e}^{\left(\frac{1}{2}\right)}(x-1) + \frac{1}{8}\mathrm{e}^{\left(\frac{1}{2}\right)}(x-1)^2 + \frac{1}{48}\mathrm{e}^{\left(\frac{1}{2}\right)}(x-1)^3$$

```
> convert(taylor(f,x=x0,6), polynom);
```

$$\mathrm{e}^{\left(\frac{1}{2}\right)}+\frac{1}{2}\mathrm{e}^{\left(\frac{1}{2}\right)}(x-1)+\frac{1}{8}\mathrm{e}^{\left(\frac{1}{2}\right)}(x-1)^2+\frac{1}{48}\mathrm{e}^{\left(\frac{1}{2}\right)}(x-1)^3+\frac{1}{384}\mathrm{e}^{\left(\frac{1}{2}\right)}(x-1)^4+\frac{1}{3840}\mathrm{e}^{\left(\frac{1}{2}\right)}(x-1)^5$$

□

Pro získání konkrétního koeficientu v Taylorově rozvoji slouží procedura `coeftayl`.

```
> coeftayl(f,x=1,3);
```

$$\frac{1}{48}\mathrm{e}^{\left(\frac{1}{2}\right)}$$

Příklad 3.9. *Určete Taylorův polynom stupně deset funkce* $y = \mathrm{e}^{x^2}$ *se středem v bodě* $x_0 = 0$.

Řešení.

```
> f:=exp(-x^2):
> convert(taylor(f,x,11), polynom);
```

$$1 - x^2 + \frac{1}{2}x^4 - \frac{1}{6}x^6 + \frac{1}{24}x^8 - \frac{1}{120}x^{10}$$

□

Pomocí procedury `taylor` však nezískáme Taylorovu řadu v uzavřeném tvaru (3.3). Tato úloha je automaticky obtížně řešitelná. Přesto v programu Maple existuje několik způsobů, jak rozvinout danou funkci do mocninné řady v uzavřeném tvaru alespoň pro některé typy funkcí. Možné přístupy budeme ilustrovat v následujících příkladech.

Příklad 3.10. *Rozviňte funkci* $y = \mathrm{e}^{2x}$ *do mocninné řady se středem v bodě* $x = 0$.

Řešení. Jednou z možností, jak tuto úlohu řešit, je využít proceduru `convert` s parametrem `Sum`. Tento příkaz vrací reprezentaci funkce pomocí nekonečné funkční řady.[4]

```
> convert(exp(2*x),Sum);
```

$$\sum_{\_k1=0}^{\infty} \frac{(2x)^{\_k1}}{\_k1!}$$

Pomocí substituce provedeme již jen kosmetickou úpravu.

[4] Více informací nalezneme v nápovědě programu Maple pod heslem `convert/Sum`.

```
> subs(_k1=k,%);
```

$$\sum_{k=0}^{\infty} \frac{(2x)^k}{k!}$$

Řadu v tomto tvaru již můžeme využít v dalších výpočtech, například při výpočtu oboru konvergence.

```
> an:=(n,x)->(2*x)^n/n!:
```

$$an := (n, x) \to \frac{(2x)^n}{n!}$$

```
> TestSeriesConvergence(an(n,x),n,x);
```

$$real$$

Získaný rozvoj tedy konverguje v celém $\mathbb{R}$. □

Příklad 3.11. *Rozviňte funkci* $y = \ln(1 + x)$ *do mocninné řady se středem v bodě* $x = 0$.

Řešení. Budeme postupovat jako v předchozím případě.

```
> convert(ln(1+x), Sum);
```

$$\ln(1 + x)$$

Tentokrát jsme nedostali hledanou odpověď. V tomto případě je důvodem fakt, že hledaná mocninná řada nekonverguje na celém definičním oboru funkce $\ln(1 + x)$ a proto provedení konverze selhalo.

Další možností k získání hledaného rozvoje je využití procedury `FunctionAdvisor` s parametrem `sum_form`.[5]

```
> FunctionAdvisor(sum_form, ln(x+1));
```

$$\left[\ln(x+1) = x\left(\sum_{\_k1=0}^{\infty} \frac{(-x)^{\_k1}}{1+\_k1}\right), And(|x| < 1)\right]$$

Ve výstupu procedury nalezneme hledanou mocninnou řadu i s podmínkou omezující platnost uvedené rovnosti na x z intervalu $(-1, 1)$. Omezíme-li se na tento interval, obdržíme mocninnou řadu také pomocí procedury `convert`.

```
> assume(x::RealRange(Open(-1),Open(1)));
> convert(ln(x+1),Sum);
```

$$x\left(\sum_{\_k1=0}^{\infty} \frac{(-x)^{\_k1}}{1+\_k1}\right)$$

Ve verzi Maple 11 je v proceduře `convert` k dispozici nový parametr `FormalPowerSeries`. Jeho uvedením získáme rozvoj i v případě, kdy mocninná řada nekonverguje na celém definičním oboru zadané funkce.

[5] Více informací o proceduře je k dispozici v nápovědě po zadání příkazu `?FunctionAdvisor`.

```
> convert(ln(1+x), FormalPowerSeries, x);
```

$$\sum_{k=0}^{\infty} \frac{(-1)^k x^{(k+1)}}{k+1}$$

Uvedená mocninná řada je ve skutečnosti konvergentní na intervalu $(-1, 1]$, přesvědčit se o tom můžeme například pomocí procedury `TestSeriesConvergence`.

```
> an:=(n,x)->x*(-x)^n/(1+n);
```

$$an := (n, x) \rightarrow \frac{x(-x)^n}{1+n}$$

```
> TestSeriesConvergence(an(n,x) ,n,x);
```

$$RealRange(Open(-1), 1)$$

□

Ve verzi Maple 11 jsou v proceduře `convert` k dispozici i další nástroje, které můžeme při výpočtu využít. Například lze pevně zvolit střed hledaného rozvoje. Nepovinný parametr `dummy` zase určuje neznámou, která bude použita pro index nekonečného součtu.[6]

Příklad 3.12. *Rozviňte funkci $y = \mathrm{e}^{2x}$ do mocninné řady se středem v bodě $x = 1$.*

Řešení.

```
> convert(exp(2*x), Sum, x=1, dummy=k);
```

$$\sum_{k=0}^{\infty} \frac{\mathrm{e}^2 2^k (x-1)^k}{k!}$$

□

3.4 Knihovna `powseries`

Konverze částečného součtu mocninné řady na polynom nám na jednu stranu umožní jeho využití v dalších výpočtech, na straně druhé se musíme omezit na polynomy konkrétního stupně a rezignovat tak na manipulaci s nekonečnou řadou jako celkem. Zároveň tím již předem limitujeme přesnost výpočtu.

V této části si představíme standardní knihovnu `powseries`, která obsahuje několik procedur pro manipulaci s mocninnými řadami.

```
> with(powseries);
  [compose, evalpow, inverse, multconst, multiply, negative, powadd,
  powcos, powcreate, powdiff, powexp, powint, powlog, powpoly, powsin,
  powsolve, powsqrt, quotient, reversion, subtract, template, tpsform]
```

[6] Obejdeme se tak bez kosmetické úpravy provedené v jednom z předchozích příkladů.

Procedura `powcreate`

Procedura `powcreate` slouží k sestrojení mocninné řady. Ty jsou v knihovně `powseries` reprezentovány prostřednictvím procedur. Parametrem procedury `powcreate` je formule popisující hodnoty koeficientů mocninné řady (tato formule může být i rekurentní) a dále pak počáteční podmínky, respektive hodnoty konkrétních koeficientů.[7]

Následující příkaz zavádí proceduru `S1`, jež je reprezentací mocninné řady

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \cdots + \frac{x^n}{n!} + \ldots,$$

tj. rozvoje funkce $y = \mathrm{e}^x$.

```
> powcreate(S1(n)=1/n!,S1(0)=1);
```

Prostřednictvím procedury `S1` můžeme přistupovat k jednotlivým koeficientům mocninné řady.

```
> S1(3);
```

$$\frac{1}{6}$$

Obecný předpis pro koeficienty mocninné řady je dostupný pomocí speciálního parametru `_k`.

```
> S1(_k);
```

$$\frac{1}{\_k!}$$

Procedura `tpsform`

Procedura `tpsform` slouží k výpisu částečného součtu mocninné řady.

```
> tpsform(S1,x);
```
[8]

$$1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5 + O(x^6)$$

Délku rozvoje lze určit pomocí třetího parametru. Není-li parametr uveden, je počet členů v rozvoji určen hodnotou proměnné prostředí `Order`.

```
> tpsform(S1,x,10);
```

$$1+x+\frac{1}{2}x^2+\frac{1}{6}x^3+\frac{1}{24}x^4+\frac{1}{120}x^5+\frac{1}{720}x^6+\frac{1}{5040}x^7+\frac{1}{40320}x^8+\frac{1}{362880}x^9+O(x^{10})$$

[7] Tyto konkrétní hodnoty pak mají přednost před obecnou formulí.

[8] Výstup příkazu je totožný s výstupem příkazu `taylor(exp(x),x);`.

Další možnosti vytvoření rozvoje

Kromě procedury `powcreate` lze k sestrojení rozvoje použít i procedury `powcos`, `powsin`, `powexp`, `powlog`, `powpoly` a `powsqrt`. Například procedura `powcos(arg)` sestrojí reprezentaci mocninného rozvoje funkce $y = \cos(arg)$. Funkce ostatních procedur je analogická.

Příklad 3.13. *Rozviňte funkci $y = \cos 2x$ do mocninné řady se středem v bodě $x_0 = 0$.*

Řešení.

```
> S2:=powcos(2*x);
```

$$S2 := \texttt{proc}(powparm) \ldots \texttt{end proc};$$

```
> tpsform(S2,x,10);
```

$$1 - 2x^2 + \frac{2}{3}x^4 - \frac{4}{45}x^6 + \frac{2}{315}x^8 + O(x^{10})$$

□

Manipulace s mocninnými řadami

Téměř všechny zbylé procedury z knihovny `powseries` slouží k manipulaci s mocninnými řadami. Na následujících řádcích si stručně představíme pouze některé z nich.

Procedury `powadd` a `subtract` slouží k součtu respektive rozdílu dvou mocninných řad. Podobně procedury `multiply` a `quotient` počítají součin respektive podíl dvou mocninných řad.

Příklad 3.14. *Pomocí rozvojů funkcí $y = \sin x$ a $y = \cos x$ určete rozvoj do mocninné řady funkce $y = \operatorname{tg} x$.*

Řešení. Při výpočtu využijeme již zmíněných procedur `powcos`, `powsin`.

```
> Order:=8:
> S1:=powsin(x):
> tpsform(S1,x);
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + O(x^8)$$

```
> S2:=powcos(x):
> tpsform(S2,x);
```

$$1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - \frac{1}{720}x^6 + O(x^8)$$

```
> S3:=quotient(S1, S2):
> tpsform(S3,x);
```

$$x + \frac{1}{3}x^3 + \frac{2}{15}x^5 - \frac{17}{315}x^7 + O(x^8)$$

□

Výstupem procedur `powdiff` respektive `powint` je řada, která vznikne derivací respektive integrací zadané mocninné řady. Je důležité si uvědomit, že uvedené operace jsou prováděny pouze formálně bez ohledu na konvergeci použitých mocninných řad.[9]

Závěrem zmiňme ještě proceduru `powsolve`, která slouží k řešení lineárních diferenciálních rovnic, přičemž získané řešení je vráceno ve formě nekonečné mocninné řady.[10]

Příklad 3.15. *Pomocí procedury* `powsolve` *nalezněte řešení homogenní lineární diferenciální rovnice*

$$y'' - xy' - y = 0$$

ve tvaru mocninné řady.

Řešení.

```
> diff(y(x),x,x) - x*diff(y(x),x) - y(x) = 0;
```

$$\left(\frac{\mathrm{d}^2}{\mathrm{d}x^2}y(x)\right) - x\left(\frac{\mathrm{d}}{\mathrm{d}x}y(x)\right) - y(x) = 0$$

```
> S:=powsolve(%);
```

$$S := \texttt{proc}(\textit{powparm}) \ldots \texttt{end proc};$$

```
> tspform(S,x);
```

$$C0 + C1\,x + \frac{1}{2}C0\,x^2 + \frac{1}{3}C1\,x^3 + \frac{1}{8}C0\,x^4 + \frac{1}{15}C1\,x^5 + O(x^6)$$

Množina řešení lineární diferenciální rovnice tvoří vektorový prostor dimenze 2, přičemž jednotlivá řešení získáme dosazením konkrétních hodnot za parametry $C0$ a $C1$.

Pomocí již zmíněného speciálního parametru `_k` se můžeme pokusit nalézt obecný předpis pro hodnoty koeficientů mocninné řady.

```
> S(0);
```

$$C0$$

```
> S(1);
```

$$C1$$

```
> S(_k);
```

$$\frac{a(\_k - 2)}{\_k}$$

Řešení diferenciální rovnice tak můžeme zapsat ve tvaru

$$S = \sum_{k=0}^{\infty} a_k x^k,$$

kde hodnota koeficientů a_k je dána rekurentní formulí

$$a_k = \frac{a_{k-2}}{k}$$

s počátečními podmínkami $a_0 = C0$ a $a_1 = C1$. □

[9] Některé z požadovaných vlastností mocninných řad jsme zmínili v části 3.2

[10] Bližší informace o proceduře jsou dostupné prostřednictvím nápovědy po zadání příkazu `?powsolve`.

3.5 Animování rozvojů do mocninných řad

V předchozí kapitole jsme věnovali pozornost také ilustrování konvergence funkčních řad vytvářením grafů a animací. Prezentované postupy lze samozřejmě využít pro řady mocninné.

Příklad 3.16. *Vytvořte animaci znázorňující konvergenci Taylorovy řady se středem v bodě* $x_0 = 0$ *funkce* $y = \sin x$.

Řešení.

```
> f:=x->sin(x):
```

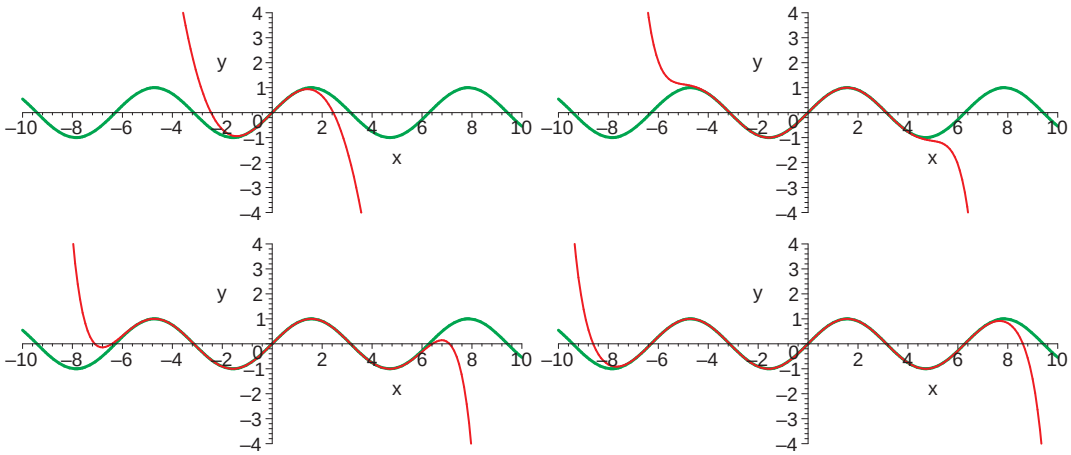
Jednotlivé Taylorovy polynomy získáme pomocí procedur `taylor` a `convert`.

```
> sn:=(n,x)->convert(taylor(f(x),x,n+1), polynom):
> sn(5,x);
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5$$

Animaci vytvoříme již známým způsobem. Pro lepší ilustraci konvergence zahrneme do animace i graf funkce $\sin x$.

```
> graf_fce:=plot(f,-10..10,thickness=2,color=green):
> graf:=n->plots[display](graf_fce, plot(sn(n,x),x=-10..10,y=-5..5,
  thickness=2)):
> plots[display](seq(graf(i),i=0..19),insequence=true,
  scaling=constrained);
```



Obr. 3.1: Ilustrace konvergence Taylorova rozvoje funkce $\sin x$ (grafy částečných součtů s_3, s_{12}, s_{15}, s_{19}).

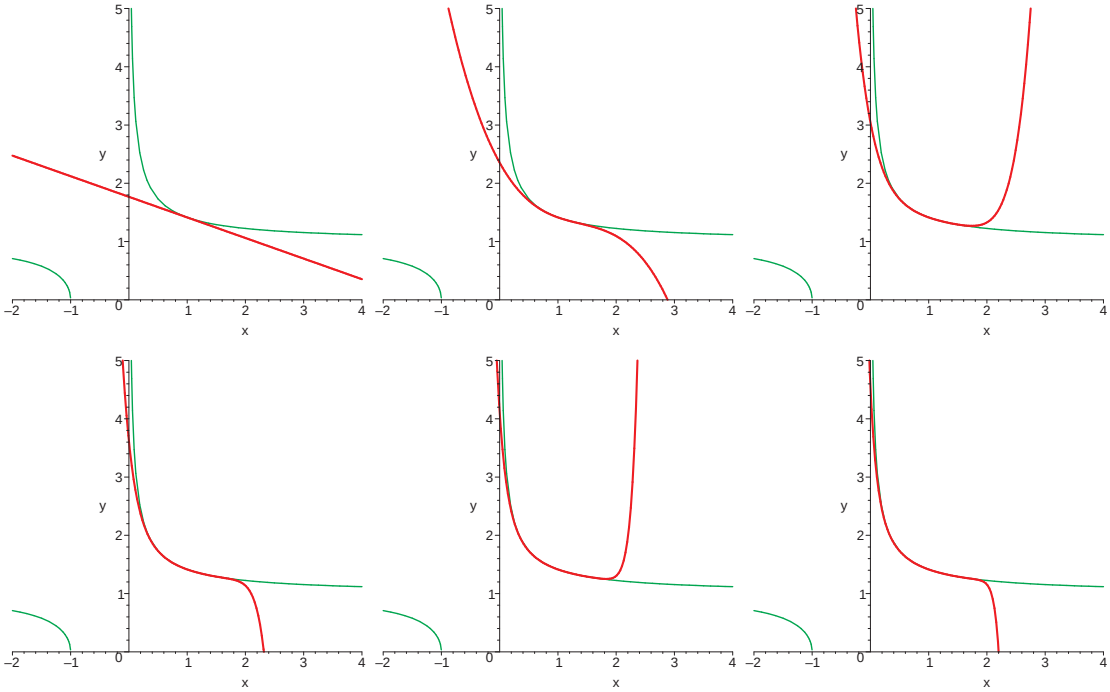
Vybrané náhledy z animace jsou na obrázku 3.1.

Taylorův rozvoj funkce $\sin x$ konverguje na celém $\mathbb{R}$. Ověřit bychom to mohli nalezením obecného předpisu pro hodnoty koeficientů a použitím procedury `TestSeriesConvergence`. □

Příklad 3.17. *Vytvořte animaci znázorňující konvergenci Taylorovy řady se středem v bodě* $x_0 = 1$ *funkce* $y = \sqrt{1 + \frac{1}{x}}$.

Řešení. Postupovat budeme jako u předchozího příkladu.

```
> f:=x->sqrt(1+1/x);
> sn:=(n,x)->convert(taylor(f(x),x=1,n+1), polynom):
> graf_fce:=plot(f,-2..4,thickness=2, color=green):
> graf:=n->plots[display](graf_fce,plot(sn(n,x),x=-2..4,y=0..5,
  thickness=2)):
> plots[display](seq(graf(i),i=0..19), insequence=true,
  scaling=constrained);
```



Obr. 3.2: Ilustrace konvergence Taylorova rozvoje funkce $y = \sqrt{1 + \frac{1}{x}}$ (grafy částečných součtů s_1, s_3, s_6, s_9, s_{12}, s_{15}).

Vybrané náhledy z animace jsou na obrázku 3.2.

Z animace je patrné, že mocninná řada má omezený obor konvergence. □

Příklad 3.18. *Vytvořte animaci znázorňující konvergenci mocninné řady*

$$\sum_{n=1}^{\infty} \frac{(-1)^{n+1} x^{n+1}}{n(n+1)}.$$

Řešení. Touto mocninnou řadou jsme se již zabývali v příkladu 3.6. Jejím oborem konvergence je interval $(-1, 1)$ a součtem funkce $y = \ln(x+1)x + \ln(x+1) - x$.

```
> fn:=(n,x)->(-1)^(n+1)*(x^(n+1))/(n*(n+1)):
> TestSeriesConvergence(fn(n,x),n,x);
```

$$RealRange(-1, 1)$$

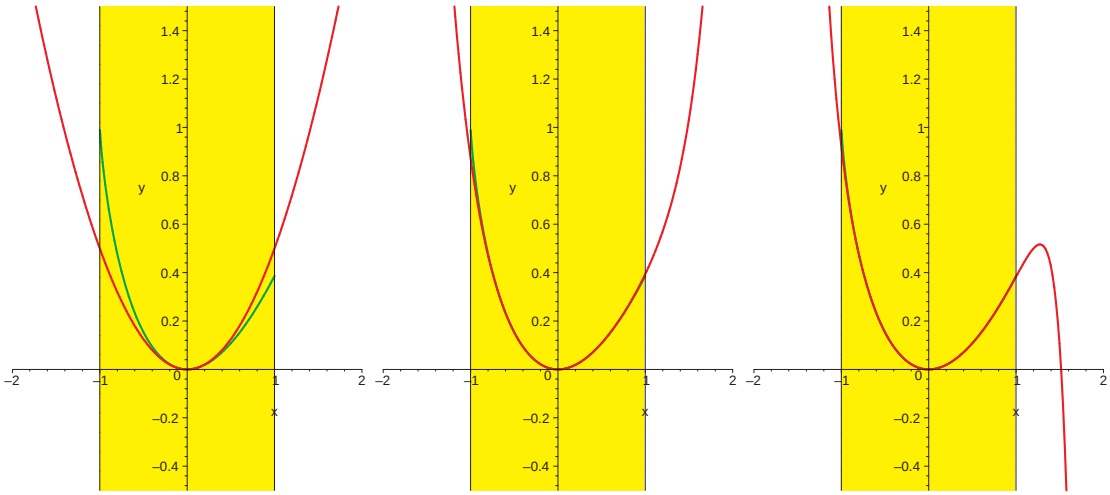
```
> s:=sum(fn(n,x),n=1..infinity);
```

$$s := \ln(x+1)x + \ln(x+1) - x$$

```
> sn:=(n,x)->sum(fn(k,x),k=1..n):
```

Obor konvergence zvýrazníme barevným pásem.

```
> graf_s:=plot(s,x=-1..1,color=green,thickness=2):
> pruh:=plotVerticalStrip(RealRange(-1,1),-0.5..1.5):
> graf:=n->plots[display](pruh, plot(sn(n,x),x=-2..2, y=-0.5..1.5,
  thickness=2),graf_s):
> plots[display](seq(graf(i),i=1..15),insequence=true,
  scaling=constrained);
```



Obr. 3.3: Ilustrace konvergence mocninné řady (grafy částečných součtů s_1, s_7, s_{12}).

Vybrané náhledy z animace jsou na obrázku 3.3. □

3.6 Užití mocninných řad

S jednou aplikací mocninných řad se v textu neustále setkáváme a to s využitím částečného součtu mocninné řady pro aproximaci funkce (samozřejmě v rámci oboru konvergence této řady). V programu Maple pravděpodobně využijeme spíše proceduru `evalf`, přesto uveďme dva krátké příklady věnované přibližnému výpočtu funkčních hodnot a určitého integrálu transcendentní funkce.

Příklad 3.19. *Pomocí Maclaurinova rozvoje funkce* $y = \operatorname{arctg} x$ *určete přibližné hodnoty* $\operatorname{arctg} \frac{1}{2}$ *a* $\operatorname{arctg} \frac{9}{10}$.

Řešení. Maclaurinova řada funkce $y = \operatorname{arctg} \mathrm{x}$ má velmi jednoduchý tvar, který můžeme odvodit ze zkráceného tvaru získaného procedurou `taylor`. Dále již snadno určíme obor konvergence $[-1, 1]$.

```
> p:=convert(taylor(arctan(x),x=0,11), polynom);
```

$$p := x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \frac{1}{9}x^9$$

Vezmeme-li v úvahu pouze nenulové členy, tak se pro pevně zvolené $x \in [-1, 1]$ jedná o alternující řadu, jejíž členy v absolutní hodnotě tvoří klesající posloupnost. Chyba, jíž se při aproximaci dopustíme, je tedy shora ohraničená dalším nenulovým členem v této posloupnosti.

```
> r:=abs(coeftayl(arctan(x),x=0,11)*x^11);
```

$$r := \frac{1}{11}|x|^{11}$$

Vyhodnotíme oba výrazy pro $x = \frac{1}{2}$.

```
> evalf(subs(x=1/2, [p, r]));
```

$$0.4636842758, 0.00004438920455$$

Tedy $\operatorname{arctg} \frac{1}{2} \approx 0,4636842758$, přičemž chyba je menší jak $5 \cdot 10^{-5}$. Výsledek můžeme ověřit pomocí procedury `evalf`.

```
> evalf(arctan(1/2));
```

$$0.4636476090$$

Stejně postupujeme pro $x = \frac{9}{10}$.

```
> evalf(subs(x=9/10, [p, r]));
```

$$0.7498165924, 0.02852823601$$

```
> evalf(arctan(9/10));
```

$$0.7328151018$$

Z prvních pěti nenulových členů Taylorova rozvoje jsme získali přibližný výsledek $\operatorname{arctg} \frac{9}{10} \approx 0,7498165924$, přičemž chyba je menší jak $3 \cdot 10^{-2}$. Z výsledku je patrné, že dosažená přesnost se od předchozího případu rapidně liší. To proto, že bod $x = \frac{9}{10}$ je blíže k hranici oboru konvergence. Tento faktor velmi výrazně ovlivňuje rychlost konvergence získané číselné řady. Vraťme se k obrázku 3.1, který zachycuje grafy několika částečných součtů Taylorova rozvoje funkce $y = \sin x$. Rozvoj konverguje na celém $\mathbb{R}$, přitom je zřejmé, že bude-li x nabývat velkých hodnot, bude třeba k dosažení rozumné přesnosti použít částečný součet velmi vysokého stupně. □

Příklad 3.20. *Pomocí Maclaurinova rozvoje funkce* $y = \mathrm{e}^{-\frac{x^2}{2}}$ *určete přibližnou hodnotu integrálu*

$$\int_{\frac{1}{2}}^{1} \mathrm{e}^{-\frac{x^2}{2}}\,\mathrm{d}x.$$

Řešení. Tentokrát využijeme procedury knihovny `powseries`. Nejdříve spočítáme Maclaurinův rozvoj funkce $y = \mathrm{e}^{-\frac{x^2}{2}}$.

```
> S:=powexp(-x^2/2):
```

Následně integrujeme řadu člen po členu.

```
> S2:=powint(S):
```

Pomocí částečného součtu určíme přibližnou hodnotu integrálu.

```
> p:=convert(tpsform(S2,x,12), polynom);
```

$$p := x - \frac{1}{6}x^3 + \frac{1}{40}x^5 - \frac{1}{336}x^7 + \frac{1}{3456}x^9$$

```
> evalf(subs(x=1,p)-subs(x=1/2,p));
```

$$.3757212644$$

Rozsah chyby omezíme shora podobně jako u předchozího příkladu.

```
> r:=abs(S2(11)*x^11);
```

$$r := \frac{1}{42240}|x|^{11}$$

```
> evalf(subs(x=1,r)+subs(x=1/2,r));
```

$$.00002368580211$$

Celkem tedy

$$\int_{\frac{1}{2}}^{1} \mathrm{e}^{-\frac{x^2}{2}}\,\mathrm{d}x \approx 0,3757212644,$$

přičemž chyba je menší než $3{\cdot}10^{-5}$. Výsledek můžeme opět ověřit pomocí procedury `evalf`.

```
> evalf(int(exp(-x^2/2),x=1/2..1));
```

$$.3756991727$$

□

Řešení diferenciálních rovnic

Nekonečné řady můžeme použít v případech, kdy nelze danou diferenciální rovnici řešit jinými standardními metodami. Používají se zejména Fourierovy řady, mocninné (Taylorovy) řady nebo zobecněné Frobeniovy řady. Řešení rovnice pak získáme ve tvaru nekonečné řady nebo jej alespoň můžeme aproximovat částečnými součty.

K řešení diferenciálních rovnic poskytuje Maple standardní proceduru `dsolve`. Pro získání řešení ve tvaru mocninné řady slouží parametr `formal_series`. Tento parametr ale lze použít pouze u homogenních lineárních obyčejných diferenciálních rovnic s polynomiálními koeficienty.

Příklad 3.21. *Najděte řešení diferenciální rovnice*

$$(3x^2 - 6x + 3)y'' + (12x - 12)y' + 6y = 0$$

ve tvaru mocninné řady.

Řešení.

```
> ode := (3*x^2-6*x+3)*diff(y(x),x$2) + (12*x-12)*diff(y(x),x)+6*y(x)=0;
```

$$ode := \left(3x^2 - 6x + 3\right)\left(\frac{\mathrm{d}^2}{\mathrm{d}x^2}y(x)\right) + (12x - 12)\left(\frac{\mathrm{d}}{\mathrm{d}x}y(x)\right) + 6y(x) = 0$$

```
> dsolve(ode,y(x),formal_series);
```

$$y(x) = \sum_{n=0}^{\infty} \frac{(\_C1 + n\_C2)\,\Gamma(n+1)x^n}{n!}$$

```
> convert(%,factorial);
```

$$y(x) = \sum_{n=0}^{\infty} (\_C1 + n\_C2)\,x^n$$

□

U ostatních typů rovnic můžeme získat pouze aproximaci řešení pomocí částečného součtu. K tomu slouží parametr `series` respektive volba `type=series`.

Příklad 3.22. *V okolí bodu $x_0 = 0$ aproximujte řešení počáteční úlohy*

$$y' - \sin(x)y = 0, \quad y(0) = 1$$

polynomem osmého stupně.

Řešení.

```
> ode:=Diff(y(x),x)-sin(x)*y(x)=0;
```

$$ode := \left(\frac{\mathrm{d}}{\mathrm{d}x}y(x)\right) - \sin(x)y(x) = 0$$

```
> cond:=y(0)=1:
> dsolve(ode, cond, y(x), series);
```

$$y(x) = 1 + \frac{1}{2}x^2 + \frac{1}{12}x^4 + \mathrm{O}(x^6)$$

Stupeň rozvoje můžeme ovlivňovat proměnnou prostředí `Order`.

```
> Order:=10:
> dsolve(ode, cond, y(x), series);
```

$$y(x) = 1 + \frac{1}{2}x^2 + \frac{1}{12}x^4 + \frac{1}{720}x^6 + \frac{43}{40320}x^8 + \mathrm{O}(x^{10})$$

Získané řešení převedeme na polynom příkazem `convert`.

```
> convert(%, polynom);
```

$$y(x) = 1 + \frac{1}{2}x^2 + \frac{1}{12}x^4 + \frac{1}{720}x^6 + \frac{43}{40320}x^8$$

□

V následujících odstavcích stručně ilustrujeme některé postupy používané při řešení diferenciálních rovnic s použitím mocninných řad. Tyto postupy jsou používané také v proceduře `dsolve`, ale jsou před očima uživatele skryty. Aplikace těchto metod je podmíněna splněním podmínek týkajících se tvaru diferenciální rovnice, existence či jednoznačnosti jejího řešení a existence rozvoje řešení do mocninné řady. Zde se však jimi nebudeme blíže zabývat.

Metoda neurčitých koeficientů

Příklad 3.23. *Řešte diferenciální rovnici*

$$y' - 2y = 0$$

v okolí bodu $x_0 = 0$.

Řešení.

```
> ode:=diff(y(x),x)-2*y(x)=0;
```

$$ode := \left(\frac{\mathrm{d}}{\mathrm{d}x}y(x)\right) - 2y(x) = 0$$

Řešení očekáváme ve tvaru

```
> form:=y(x)=Sum(a(k)*x^k,k=0..infinity);
```

$$form := y(x) = \sum_{k=0}^{\infty} a(k)x^k$$

Dosadíme tento výraz do řešené diferenciální rovnice a upravíme.

```
> eval(ode, form);
```

$$\left(\sum_{k=0}^{\infty} \frac{a(k)x^k k}{x}\right) - 2\left(\sum_{k=0}^{\infty} a(k)x^k\right) = 0$$

```
> combine(%);
```

$$\sum_{k=0}^{\infty} \left(a(k)k \;\; x^{(k-1)} - 2a(k)x^k \right) = 0$$

Jelikož Maple nedovede manipulovat s nekonečnými řadami do té míry, aby našel obecný vztah mezi koeficienty, je nutné přejít k součtům konečným. Jelikož řešíme diferenciální rovnici prvního řádu, ve výše uvedeném vztahu se vyskytují pouze dva sousední koeficienty. To znamená, že tři členy v konečném součtu budou dostatečné pro odvození rekurentní formule.

```
> tform:=y(x)=Sum(a(n)*x^n,n=k-1..k+1);
```

$$tform := y(x) = \sum_{n=k-1}^{k+1} a(n)x^n$$

Tento výraz opět dosadíme do rovnice a upravíme.

```
> eval(ode, tform):
> q:=simplify(combine(value(%)));
```

$$q := x^{(k-2)}a(k-1)k - x^{(k-2)}a(k-1) + a(k)kx^{(k-1)} + x^k a(k+1)k + x^k a(k+1) - \\ -2a(k-1)x^{(k-1)} - 2a(k)x^k - 2a(k+1)x^{(k+1)} = 0$$

Rekurentní vztah získáme z koeficientu u mocniny x^k.

```
> req:=map(coeff,q,x^k);
```

$$req := a(k+1)k + a(k+1) - 2a(k) = 0$$

Prostřednictvím procedury `rsolve` se můžeme pokusit najít obecné řešení této rekurentní formule.

```
> rsol:=rsolve(req,{a});
```

$$rsol := \left\{ a(k) = \frac{2^k a(0)}{\Gamma(k+1)} \right\}$$

Hodnotu funkce Γ vyjádříme pomocí faktoriálu a dosadíme do tvaru hledaného řešení.

```
> rsol2:=convert(rsol, factorial):
> sol:=subs(rsol2,form);
```

$$sol := y(x) = \sum_{k=0}^{\infty} \frac{2^k a(0)x^k}{k!}$$

Z důvodu jednoduchosti diferenciální rovnice můžeme dokonce určit funkci reprezentovanou touto mocninnou řadou.

```
> value(sol);
```

$$y(x) = a(0)\mathrm{e}^{(2x)}$$

Toto řešení je ekvivalentní řešení získanému procedurou `dsolve`.

```
> dsolve(ode,y(x));
```

$$y(x) = \_C1\,\mathrm{e}^{(2x)}$$

Často však není možné získat obecné řešení rekurentní formule. V tom případě je možné zkonstruovat konečnou soustavu rovnic a z jejího řešení sestrojit aproximaci řešení původní diferenciální rovnice.

```
> N:=5:
> eqset:={req$k=0..(N-1)};
```

$$eqset := \{a(1) - 2a(0) = 0,\ \ 2a(2) - 2a(1) = 0,\ \ 3a(3) - 2a(2) = 0,\\ 4a(4) - 2a(3) = 0,\ \ 5a(5) - 2a(4) = 0\}$$

```
> solve(eqset,{a(k)$k=1..N}):
> eval(value(subs(infinity=N,form)),%);
> collect(%,a(0));
```

$$y(x) = \left(1 + 2x + 2x^2 + \frac{4}{3}x^3 + \frac{2}{3}x^4 + \frac{4}{15}x^5\right) a(0)$$

Pro úplnost poznamenejme, že získaná řešení obdržíme také při použití procedury `dsolve`.

```
> dsolve(ode,y(x),formal_series):
```

$$y(x) = \_C1 \left(\sum_{n=0}^{\infty} \frac{2^n x^n}{n!}\right)$$

```
> dsolve(ode,y(x),series):
> convert(%, polynom);
```

$$y(x) = y(0) + 2y(0)x + 2y(0)x^2 + \frac{4}{3}y(0)x^3 + \frac{2}{3}y(0)x^4 + \frac{4}{15}y(0)x^5$$

□

Metoda derivování

Příklad 3.24. *Aproximujte řešení počáteční úlohy*

$$y' = \sin(x + y), \quad y(0) = 0$$

v okolí bodu $x_0 = 0$ *Taylorovým polynomem pátého stupně.*

Řešení.

```
> ode:=Diff(y(x),x)=sin(x+y(x));
```

$$ode := \frac{\mathrm{d}}{\mathrm{d}x} y(x) = \sin(x + y(x))$$

```
> x0:=0:
> cond:=y(x0)=0:
> N:=5:
```

Řešení budeme hledat v následujícím tvaru

```
> form:=y(x)=convert(series(y(x),x=x0,N+1),polynom);
```

$$form := y(x) = y(0) + D(y)(0)x + \tfrac{1}{2}(D^{(2)})(y)(0)x^2 + \tfrac{1}{6}(D^{(3)})(y)(0)x^3 + \\ + \tfrac{1}{24}(D^{(4)})(y)(0)x^4 + \tfrac{1}{120}(D^{(5)})(y)(0)x^5$$

Funkční hodnotu v bodě 0 známe z počáteční podmínky. Pro sestrojení polynomu potřebujeme dále znát hodnoty derivací do řádu 5 v bodě 0. Tyto hodnoty budeme ukládat do proměnných `Dy[n]`.

```
> Dy[0]:=cond;
```

$$Dy_0 := y(0) = 0$$

Do proměnných `Dode[n]` budeme ukládat diferenciální rovnice řádu n získané derivováním rovnice původní. Hodnotu první derivace získáme dosazením počáteční podmínky do řešené diferenciální rovnice.

```
> Dode[1]:=convert(ode,D):
> Dy[1]:=isolate(eval(subs(x=x0, cond, Dode[1])), D(y)(x0));
```

$$Dy_1 := D(y)(0) = 0$$

Hodnotu druhé derivace získáme derivováním řešené rovnice a dosazením hodnot derivací nižších řádů.

```
> Dode[2]:=convert(diff(ode,x),D):
```

$$Dode_2 := (D^{(2)})(y)(x) = \cos(x + y(x))(1 + D(y)(x))$$

```
> Dy[2]:=isolate(eval(subs(x=x0, Dy[i]$i=0..1, Dode[2])), (D@@2)(y)(x0));
```

$$Dy_2 := (D^{(2)})(y)(0) = 1$$

Opakováním tohoto procesu získáme hodnoty derivací vyšších řádů. My jej automatizujeme s využitím cyklu.

```
> for n from 3 to N do
> Dode[n]:=simplify(convert(diff(Dode[n-1],x),D));
> Dy[n]:=isolate(eval(subs(x=x0, Dy[i]$i=0..(n-1), Dode[n])),
  (D@@(n))(y)(x0));
> end do:
```

Získali jsme následující hodnoty derivací.

```
> print(Dy[i]$i=0..N);
```

$$y(0) = 0, D(y)(0) = 0, (D^{(2)})(y)(0) = 1, (D^{(3)})(y)(0) = 1, \\ (D^{(4)})(y)(0) = 0, (D^{(5)})(y)(0) = -6$$

Nyní již můžeme sestrojit hledaný polynom.

```
> sol:=eval(form, {Dy[i]$i=0..N});
```

$$sol := y(x) = \frac{1}{2}x^2 + \frac{1}{6}x^3 - \frac{1}{20}x^5$$

Námi uváděný příklad byl modelový, stejný výsledek získáme procedurou `dsolve` následovně.

```
> Order:=6:
> convert(dsolve({ode, cond}, y(x), type=series), polynom);
```

$$y(x) = \frac{1}{2}x^2 + \frac{1}{6}x^3 - \frac{1}{20}x^5$$

□

Kapitola 4

Fourierovy trigonometrické řady

V následujících dvou kapitolách se budeme věnovat Fourierových řadám. Podobně jako Taylorovy řady z předchozí kapitoly, také Fourierovy řady představují rozvoj dané funkce do funkční řady. V případě Fourierových trigonometrických řad hledáme vyjádření pomocí lineární kombinace funkcí

$$1, \cos x, \sin x, \cos 2x, \sin 2x, \cos 3x, \sin 3x, \ldots, \tag{4.1}$$

tedy v případě konečného rozvoje tzv. *trigonometrický polynom*[1] ve tvaru

$$T_n(x) = a_0 + \sum_{k=1}^{n} (a_k \cos kx + b_k \sin kx), \quad a_0, a_k, b_k \in \mathbb{R}, \tag{4.2}$$

nebo jako nekonečnou *trigonometrickou řadu*

$$a_0 + \sum_{n=1}^{\infty} (a_n \cos nx + b_n \sin nx). \tag{4.3}$$

Jedná se o 2π-periodické funkce, jsou proto vhodné právě pro aproximaci 2π-periodických funkcí.

V případě Fourierových řad hraje důležitou roli *skalární součin* a *ortogonalita* zvoleného systému funkcí.

Definice. Buďte f, g integrovatelné funkce na intervalu $[a, b]$. Číslo

$$(f,\, g) = \int_a^b f(x) g(x)\, \mathrm{d}x$$

nazýváme *skalárním součinem* funkcí f, g. Funkce f, g se nazývají *ortogonální* (na intervalu $[a, b]$), právě když $(f,\, g) = 0$.

[1] Název polynom je odůvodněn tím, že užitím elementárních vztahů z trigonometrie lze $T_n(x)$ vyjádřit jako polynom v proměnných $\cos x, \sin x$.

Definice. Buď f funkce integrovatelná na intervalu $[a,b]$. *Normou funkce* f rozumíme číslo $\|f\| = \sqrt{(f,f)}$. Funkce f se nazývá *normovaná*, právě když $\|f\| = 1$.

Definice. Buď $\{\varphi_n\}$ konečná nebo spočetná posloupnost integrovatelných funkcí na intervalu $[a,b]$. Tato posloupnost se nazývá *ortogonální*, právě když každé dvě funkce φ_m, φ_n $(m \neq n)$ jsou ortogonální a každá funkce φ_n má kladnou normu.
Posloupnost $\{\varphi_n\}$ se nazývá *ortonormální*, právě když je ortogonální a každá funkce φ_n je normovaná.

Posloupnost funkcí (4.1) tvoří ortogonální systém na intervalu $[0, 2\pi]$, respektive na každém intervalu tvaru $[c, c+2\pi]$, kde $c \in \mathbb{R}$. V této kapitole se budeme věnovat Fourierovým řadám právě vzhledem k systému (4.1). Existují i další systémy funkcí, některým z nich je věnována následující kapitola.

K výpočtu Fourierových řad budeme využívat následující větu.

Věta. Fourierova řada libovolné integrovatelné funkce f na intervalu $[-\pi, \pi]$ má vzhledem k systému (4.1) tvar

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos nx + b_n \sin nx), \tag{4.4}$$

kde a_n, b_n jsou Fourierovy koeficienty funkce f, pro něž platí

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

S ohledem na periodicitu funkcí v systému (4.1) lze výše uvedené úvahy beze zbytku převést z intervalu $[-\pi, \pi]$ na libovolný interval $[c, c+2\pi]$, $c \in \mathbb{R}$.

Poznámka. Fourierovu řadu (4.4) lze vyjádřit v oboru $\mathbb{C}$ užitím vztahů

$$\cos x = \frac{\mathrm{e}^{ix} + \mathrm{e}^{-ix}}{2}, \quad \sin x = \frac{\mathrm{e}^{ix} - \mathrm{e}^{-ix}}{2i}$$

takto:

$$\frac{a_0}{2} + \frac{1}{2} \sum_{n=1}^{\infty} \left(a_n (\mathrm{e}^{inx} + \mathrm{e}^{-inx}) - b_n i (\mathrm{e}^{inx} - \mathrm{e}^{-inx}) \right) =$$

$$c_0 + \sum_{n=1}^{\infty} \left(\frac{a_n - b_n i}{2} \mathrm{e}^{inx} + \frac{a_n + b_n i}{2} \mathrm{e}^{-inx} \right) = \sum_{n=-\infty}^{\infty} c_n \mathrm{e}^{inx},$$

kde Fourierovy koeficienty c_n jsou tvaru

$$c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) \mathrm{e}^{-inx} \, \mathrm{d}x, \qquad n = 0, \pm 1, \pm 2, \cdots .$$

4.1 Výpočet metodou „krok za krokem“

V této a následujících kapitolách se budeme věnovat tématu Fourierových řad s využitím programu Maple. Nejdříve budeme postupovat „krok za krokem“, jako bychom výpočet prováděli ručně. Tento postup je sice pomalejší, avšak názornější a umožňuje nám, je-li to vhodné, Maplu při výpočtu asistovat.

Příklad 4.1. *Spočítejte Fourierovu řadu funkce $f(x) = x^2$ definované na intervalu $[-\pi, \pi]$ a pomocí animace znázorněte konvergenci jejích částečných součtů.*

Řešení. Pro počítání koeficientů a_n a b_n budeme potřebovat celočíselnou proměnnou `n`.

```
> restart;
> assume(n,integer);
```

Nyní spočtěme koeficienty a_0, a_n a b_n. Využijeme dříve uvedených vztahů.

```
> a0:=1/Pi*int(x^2, x=-Pi..Pi);
```

$$a0 := \frac{2\pi^2}{3}$$

```
> aN:=1/Pi*int(x^2*cos(n*x), x=-Pi..Pi);
```

$$aN := \frac{4(-1)^n}{n^2}$$

Protože je x^2 funkce sudá, bude koeficient b_n roven nule a Fourierova řada tak nebude obsahovat sinové členy. Přesto tuto skutečnost ověříme výpočtem.

```
> bN:=1/Pi*int(x^2*sin(n*x), x=-Pi..Pi);
```

$$bN := 0$$

Fourierova řada funkce $f(x) = x^2$ má tedy tvar

```
> a0/2+Sum(aN*cos(n*x)+bN*sin(n*x), n=1..infinity);
```

$$\frac{\pi^2}{3} + \left(\sum_{n=1}^{\infty} \left(\frac{4(-1)^n \cos(nx)}{n^2} \right) \right)$$

Při výpočtu Fourierovy řady jsme k zápisu výsledku použili proceduru `Sum`. Nesmíme ji zde zaměnit s procedurou `sum`, která slouží k výpočtu součtu řady. Maple by se snažil najít součet této řady, což by se mu sice nepodařilo a výsledek by byl stejný, zaplatili bychom za to však časovou prodlevou, která může u některých funkcí trvat i několik minut.

Nyní si znázorníme grafem jednotlivé částečné součty. Nejdříve vytvoříme funkci `four`, která pro zadané m vrací trigonometrický polynom $T_m(x)$. V tomto případě již použijeme proceduru `sum`.

```
> four:=(m,x)->a0/2+sum(aN*cos(n*x)+bN*sin(n*x), n=1..m):
```

Například trigonometrický polynom $T_3(x)$ má tvar:

```
> T[3](x)=four(3,x);
```

$$T_3(x) = \frac{\pi^2}{3} - 4\cos(x) + \cos(2x) - \frac{4}{9}\cos(3x)$$

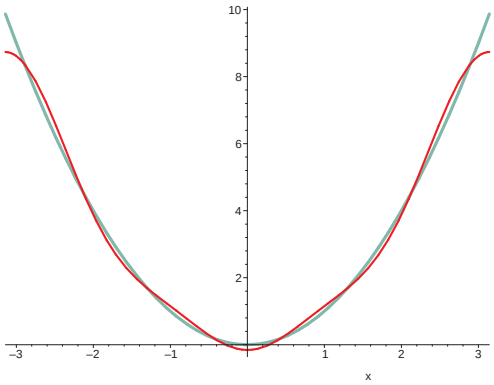
Načteme knihovnu `plots` obsahující procedury pro kreslení grafů.

```
> with(plots):
```

Do proměnné `graf1` uložíme graf funkce x^2 na intervalu $[-\pi, \pi]$, do proměnné `graf2` graf trigonometrického polynomu $T_3(x)$. Poté je společně zobrazíme pomocí příkazu `display`.

```
> graf1:=plot(x^2,x=-Pi..Pi,color=aquamarine,thickness=2):
> graf2:=plot(four(3,x),x=-Pi..Pi,color=red):
> display(graf1,graf2);
```

Výsledný graf je na obrázku 4.1.



Obr. 4.1: Funkce $f(x) = x^2$ a trigonometrický polynom $T_3(x)$.

Nyní vytvoříme animaci znázorňující přibližování Fourierovy řady k původní funkci. Při animaci použijeme prvních 10 členů.

```
> clenu:=10:
```

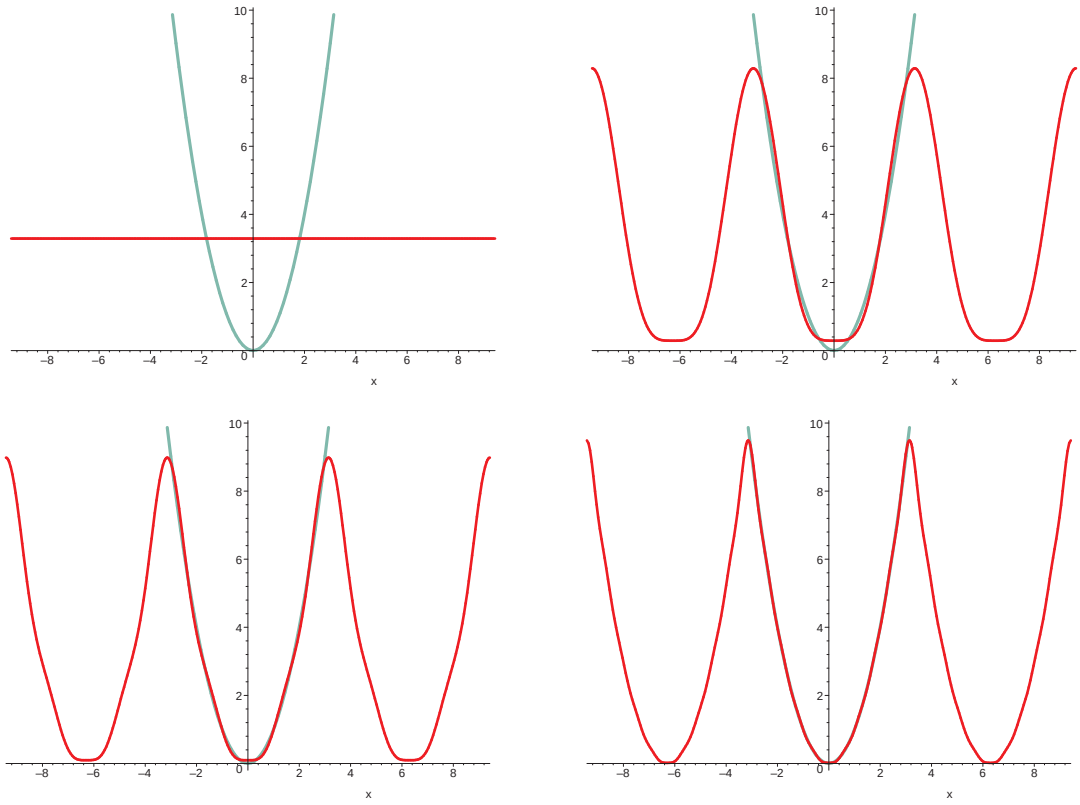
Do proměnné `anim` uložíme animaci trigonometrického polynomu $T_m(x)$ při rostoucí hodnotě m. Pro společné zobrazení spolu s grafem funkce x^2 použijeme opět příkaz `display`.

```
> anim:=animate(four(m,x),x=-3*Pi..3*Pi,m=0..clenu,color=red,
  frames=clenu+1,numpoints=300):
> display(graf1,anim);
```

Vybrané náhledy jsou na obrázku 4.2. Z animace je dobře patrná konvergence Fourierovy řady k 2π-periodickému rozšíření funkce x^2.

Detail okolí bodu nula zobrazíme například následujícím příkazem.

```
> display([graf1,anim],view=[-1.2..1.2,-0.8..1.6]);
```



Obr. 4.2: Vybrané náhledy z animace (pro m=0, 2, 4 a 10)

Jistě nás bude zajímat, jak moc se liší funkční hodnoty jednotlivých částečných součtů řady od funkčních hodnot funkce, jejíž Fourierovu řadu jsme počítali. Základní představu jsme si již udělali z animace. Zabývejme se nyní chybou aproximace, tedy rozdílem $g_n(x) = f(x) - s_n(x)$ na základním intervalu. Pomocí animace ukážeme, jak se mění průběh funkce $g_n(x)$ v závislosti na rostoucí hodnotě n.

```
> animate(x^2-four(m,x), x=-Pi..Pi, m=1..clenu, frames=clenu,
  numpoints=300, view=[-Pi..Pi,-1..1], scaling=constrained, thickness=2);
```

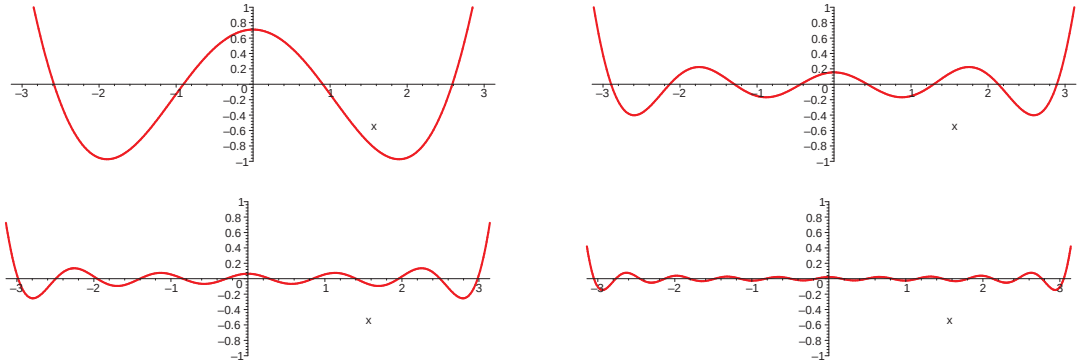
Vybrané náhledy z animace jsou na obrázku 4.3. Také z této animace je patrné přibližování funkce $g_n(x)$ k ose y, zpětně tedy i konvergence Fourierovy řady k původní funkci (stále se pohybujeme pouze na intervalu $[-\pi, \pi]$).

□

Jako kritérium přesnosti aproximace funkce částečným součtem Fourierovy řady lze použít *kvadratickou odchylku*. Kvadratickou odchylku dvou integrovatelných funkcí f a g na intervalu $[a, b]$ určíme vztahem

$$\|f - g\| = \left\{ \int_a^b [f(x) - g(x)]^2 \, dx \right\}^{1/2}. \tag{4.5}$$

Příklad 4.2. *Určete kvadratickou odchylku prvních deseti částečných součtů Fourierovy řady funkce z předcházejícího příkladu.*



Obr. 4.3: Vybrané náhledy z animace (pro n=1, 3, 5 a 9)

Řešení. Definujeme funkci na výpočet kvadratické odchylky dvou funkcí pomoci vztahu (4.5).

```
> qdev:= (f, g, var) -> evalf(sqrt(int((f-g)^2, var))):
```

Funkci `qdev` předáváme tři parametry. Prvními dvěma parametry jsou požadované funkce, třetím parametrem je neznámá spolu s vyšetřovaným intervalem.

Nyní určíme hodnotu kvadratické odchylky pro částečné součty $s_0(x)$ až $s_9(x)$.

```
> for i from 0 to 9 do
> qdev(x^2, four(i,x), x=-Pi..Pi);
> end do;
```

```
7.375872795
2.034211661
0.9982106214
0.6130765968
0.4236902112
0.3147834451
0.2455677473
0.1984145125
0.1646099789
0.1394101696
```

V případě částečných součtů vyšších řádů se již výpočet stává časově náročným. Pro $s_{50}(x)$ je kvadratická odchylka rovna 0.01140528237, pro $s_{100}(x)$ 0.004065598145 a pro $s_{200}(x)$ pak 0.001441114406. □

Vztah (4.5) není z hlediska efektivity výpočtu příliš vhodný pří práci s částečnými součty vyšších řádů. V případě kvadratické odchylky funkce a částečného součtu její Fourierovy řady je vhodnější použít *Besselovy identity*, viz následující poznámka.

Poznámka. Pro funkci f a její Fourierovu řadu platí tzv. *Besselova identita*

$$\left\| f - \sum_{k=1}^{n} c_k \varphi_k \right\|^2 = \|f\|^2 - \sum_{k=1}^{n} c_k^2 \|\varphi_k\|^2.$$

Hodnoty výrazů uvnitř tohoto vzorce (norma funkce f, hodnoty Fourierových koeficientů a normy funkcí tvořících ortogonální systém) lze v našem případě vyčíslit předem a jsou prakticky nezávislé na stupni částečného součtu. K této úloze se ještě vrátíme v části s řešenými příklady.

Příklad 4.3. *Spočítejme Fourierovu řadu funkce* $f(x) = |\sin x|$ *na intervalu* $[-\pi, \pi]$.

Řešení. Budeme postupovat stejně jako v předchozím textu.

```
> f:=x->abs(sin(x)):
> a0:=1/Pi*int(f(x), x=-Pi..Pi);
```

$$a0 := \frac{4}{\pi}$$

```
> aN:=1/Pi*int(f(x)*cos(n*x), x=-Pi..Pi);
```

$$aN := \frac{2((-1)^n + 1)}{\pi(-1 + n^2)}$$

```
> bN:=1/Pi*int(f(x)*sin(n*x), x=-Pi..Pi);
```

$$bN := 0$$

Fourierova řada by tedy měla mít tvar:

```
> a0/2+Sum(aN*cos(n*x)+bN*sin(n*x),n=1..infinity);
```

$$\frac{2}{\pi} + \sum_{n=1}^{\infty} -\frac{2((-1)^n + 1)\cos(nx)}{\pi(-1 + n^2)}$$

Všimněme si však, že pro $n = 1$ není koeficient a_1 definován.[2] Musíme tedy tento případ vyšetřit zvlášť.

```
> a1:=1/Pi*int(f(x)*cos(x), x=-Pi..Pi);
```

$$a1 := 0$$

```
> b1:=1/Pi*int(f(x)*sin(x), x=-Pi..Pi);
```

$$b1 := 0$$

Oba koeficienty jsou rovny nule, hledaná Fourierova řada má tvar

```
> a0/2+Sum(aN*cos(n*x)+bN*sin(n*x),n=2..infinity);
```

$$\frac{2}{\pi} + \sum_{n=2}^{\infty} -\frac{2((-1)^n + 1)\cos(nx)}{\pi(-1 + n^2)}$$

[2]Totéž platí i pro koeficient b_1, v následujícím příkladu uvidíme, jak je v takovém případě důležité ověřit také hodnotu druhého z koeficientů.

□

Příklad 4.4. *Určete Fourierovu řadu funkce definované předpisem*

$$f(x) = \begin{cases} 0 & pro\ x \in [-\pi, 0) \\ \sin(x) & pro\ x \in [-0, \pi] \end{cases}$$

na intervalu $[-\pi, \pi]$.

Řešení.

```
> f:=x->piecewise(x<0,0,sin(x)):
> a0:=1/Pi*int(f(x), x=-Pi..Pi);
```

$$a0 := \frac{2}{\pi}$$

```
> aN:=1/Pi*int(f(x)*cos(n*x), x=-Pi..Pi);
```

$$aN := \frac{(-1)^n + 1}{\pi(-1 + n^2)}$$

```
> bN:=1/Pi*int(f(x)*sin(n*x), x=-Pi..Pi);
```

$$bN := 0$$

Opět budeme vyšetřovat případ $n = 1$.

```
> a1:=1/Pi*int(f(x)*cos(x), x=-Pi..Pi);
```

$$a1 := 0$$

```
> b1:=1/Pi*int(f(x)*sin(x), x=-Pi..Pi);
```

$$b1 := \frac{1}{2}$$

Všimněme si, že koeficient b_1 je jako jediný koeficient u sinových členů nenulový. Fourierova řada má tvar

```
> a0/2+b1*sin(x)+Sum(aN*cos(n*x)+bN*sin(n*x),n=2..infinity);
```

$$\frac{1}{\pi} + \frac{1}{2}\sin(x) + \sum_{n=2}^{\infty} -\frac{((-1)^n + 1)\cos(nx)}{\pi(-1 + n^2)}$$

Pokud se zamyslíme nad „blízkostí“ funkce f k funkci z předcházejícího příkladu, jistě nás nepřekvapí podobnost jejích Fourierových řad. Ze znalosti tvaru jedné z řad je snadné určit také tvar řady druhé. □

Fourierova řada funkce f na intervalu $[a, b]$

Opět provedeme výpočet Fourierovy řady funkce f, neomezíme se však na interval $[-\pi, \pi]$, ale předvedeme si řešení použitelné pro obecně zadaný interval $I = [a, b]$.

Označme L délku tohoto intervalu. Pro výpočet koeficientů použijeme následující vztahy:

$$\begin{aligned} a_0 &= \frac{2}{L}\int_a^b f(x)\,\mathrm{d}x \\ a_n &= \frac{2}{L}\int_a^b f(x)\cos\left(\frac{2\pi n x}{L}\right)\,\mathrm{d}x \\ b_n &= \frac{2}{L}\int_a^b f(x)\sin\left(\frac{2\pi n x}{L}\right)\,\mathrm{d}x \end{aligned}$$

Fourierova řada pak bude mít tvar

$$\frac{a_0}{2} + \sum_{n=1}^{\infty}\left(a_n\cos\left(\frac{2\pi n x}{L}\right) + b_n\sin\left(\frac{2\pi n x}{L}\right)\right).$$

Příklad 4.5. *Spočítejte hodnoty koeficientů Fourierovy řady funkce zadané předpisem* $f(x) = 1 - x^2 + x$, $x \in [-\frac{1}{2}, 1]$ *a pomocí grafů vybraných částečných součtů ilustrujte konvergenci Fourierovy řady.*

Řešení.

```
> restart:
> assume(n, integer):
```

Nejdříve zadáme předpis funkce, krajní body a délku intervalu. Jejich uložením do proměnných získáme možnost v budoucnu snadno modifikovat zadání.

```
> f:=x->1-x^2+x:
> a:=-1/2:
> b:=1:
> L:=abs(a-b):
```

S využitím výše uvedených vztahů spočítáme koeficienty a_0, a_n a b_n.

```
> a0:=2/L*int(f(x), x=a..b);
```

$$a0 := 2$$

```
> aN:=2/L*int(f(x)*cos(2*Pi/L*n*x), x=a..b);
```

$$\begin{aligned} aN :=& \frac{1}{8}\left(-6\pi n\cos\left(\frac{4\pi n}{3}\right) + 9\sin\left(\frac{4\pi n}{3}\right) + 8\pi^2 n^2\sin\left(\frac{4\pi n}{3}\right) + 2\pi^2 n^2\sin\left(\frac{2\pi n}{3}\right) - \right. \\ & \left. - 12\pi n\cos\left(\frac{2\pi n}{3}\right) + 9\sin\left(\frac{2\pi n}{3}\right)\right) \Big/ (\pi^3 n^3) \end{aligned}$$

```
> bN:=2/L*int(f(x)*sin(2*Pi/L*n*x), x=a..b);
```

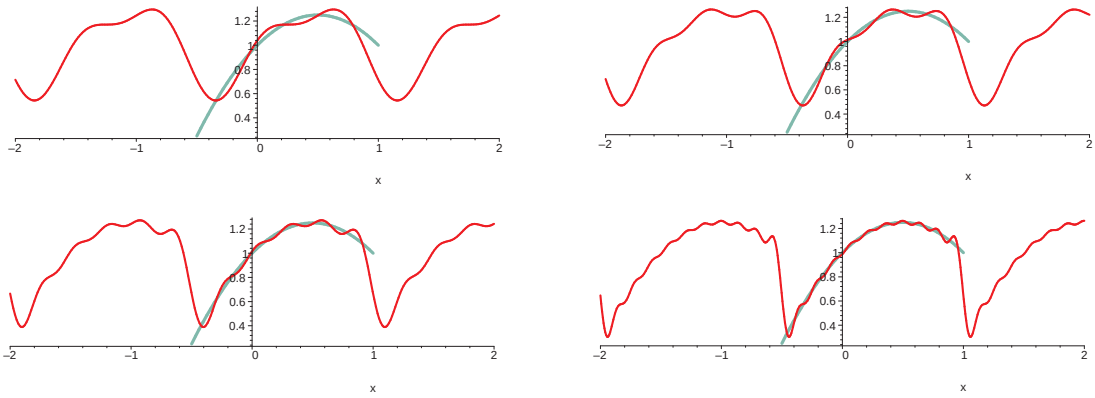
$$bN := -\frac{1}{8}\left(6\pi n \sin\left(\frac{4\pi n}{3}\right) + 9\cos\left(\frac{4\pi n}{3}\right) + 8\pi^2 n^2 \cos\left(\frac{4\pi n}{3}\right) - 2\pi^2 n^2 \cos\left(\frac{2\pi n}{3}\right) - \right.$$
$$\left. - 12\pi n \sin\left(\frac{2\pi n}{3}\right) - 9\cos\left(\frac{2\pi n}{3}\right)\right) \Big/ (\pi^3 n^3)$$

Jelikož výsledné výrazy jsou velice dlouhé, nebudeme zde uvádět celou Fourierovu řadu. Lze ji získat následujícím příkazem

```
> a0/2+Sum(aN*cos(2*Pi/L*n*x)+bN*sin(2*Pi/L*n*x),n=1..infinity);
```

Funkci vracející částečné součty Fourierovy řady zadefinujeme příkazem

```
> four:=(m,x)->a0/2+sum(aN*cos(2*Pi/L*n*x)+bN*sin(2*Pi/L*n*x), n=1..m):
```



Obr. 4.4: Vybrané náhledy z animace (pro m=2, 3, 5 a 10)

Při vytváření grafů a animací postupujeme již známým způsobem. Uvedeme pouze příkazy pro vytvoření několika náhledů z animace.

```
> with(plots):
> graf1:=plot(f(x), x=a..b, color=aquamarine, thickness=2):
> for i in [2,3,5,10] do
> display(graf1, plot(four(i,x), x=-2..2, color=red, numpoints=300),
  scaling=constrained);
> end do;
```

Výsledné grafy jsou na obrázku 4.4. □

4.2 Konvergence Fourierovy řady

Přiřazení Fourierovy řady k dané funkci f je ovšem zatím pouze formální, neboť nevíme, zda tato řada vůbec konverguje, a v případě její konvergence, zda její součet je f. Dostatečnou podmínku bodové konvergence zmiňuje Dirichletova věta.

Nazvěme funkci f *po částech spojitou* na intervalu $[a, b]$, právě když má na tomto intervalu pouze konečný počet bodů nespojitosti, přičemž tyto body jsou body nespojitosti

prvního druhu (tj. v těchto bodech existují obě jednostranné limity a jsou vlastní). Nazvěme funkci f *po částech monotónní* na intervalu $[a, b]$, právě když existuje dělení tohoto intervalu (s konečným počtem bodů) tak, že uvnitř každého dělícího intervalu je daná funkce monotónní.

Věta (Dirichletova)**.** Nechť funkce f je po částech spojitá a po částech monotónní na intervalu $[-\pi, \pi]$. Pak její Fourierova řada konverguje na $[-\pi, \pi]$ a její součet je roven:

(1) $f(x_0)$ v každém bodě $x_0 \in (-\pi, \pi)$, v němž je f spojitá,

(2) $\frac{1}{2}[f(x_0-) + f(x_0+)]$ v každém bodě $x_0 \in (-\pi, \pi)$, v němž je f nespojitá,

(3) $\frac{1}{2}[f(-\pi+) + f(\pi-)]$ v krajních bodech intervalu $[-\pi, \pi]$,

přičemž symbolem $f(x_0+)$ se rozumí číslo $\lim_{x\to x_0+} f(x)$, pokud tato jednostranná limita existuje. Analogicky je $f(x_0-) = \lim_{x\to x_0-} f(x)$.

Tvrzení předchozí věty je dobře patrné například na obrázcích 4.2 a 4.4.

Gibbsův jev

V této části se budeme krátce věnovat problému souvisejícím s konvergencí Fourierových řad nespojitých funkcí, tzv. *Gibbsově jevu*. Okolo bodů nespojitosti dochází u částečných součtů Fourierovy řady k „překmitům“, dobře je tento jev patrný na mnoha obrázcích v dalším textu, například na Obr. 4.11 nebo 4.12.

Zajímavé je, že s vyššími částečnými součty zůstává tento překmit prakticky stále stejný, dochází k němu však na stále menších intervalech. O konvergenci Fourierovy řady stále platí tvrzení uváděná v Dirichletově větě.

Označme $g(x)$ liché 2π-periodické rozšíření funkce $f(x) = x$ z intervalu $[0, \pi]$. Pro funkci g platí

$$g(x) = 2\sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} \sin(nx).$$

Nechť s_n je n-tý částečný součet uvedené řady. Pro hodnotu součtu s_n v bodě $x = \pi - \frac{\pi}{n}$ dostáváme

$$s_n\left(\pi - \frac{\pi}{n}\right) = \sum_{k=1}^{n} \frac{2}{k} \sin\left(\frac{k\pi}{n}\right) = 2\sum_{k=1}^{n} \frac{\pi}{n}\left(\frac{n}{k\pi} \sin\left(\frac{k\pi}{n}\right)\right).$$

Bližším pohledem zjistíme, že výraz ve vnější závorce představuje hodnotu funkce $\frac{\sin x}{x}$ v bodě $x = \frac{k\pi}{n}$, výraz před závorkou značí délku dělení intervalu. Tedy

$$2\sum_{k=1}^{n} \frac{\pi}{n}\left(\frac{n}{k\pi} \sin\left(\frac{k\pi}{n}\right)\right) \longrightarrow 2\int_0^{\pi} \frac{\sin x}{x}\,\mathrm{d}x \quad \text{pro } n \to \infty.$$

Hodnota výrazu na pravé straně je přibližně rovna 3.703874104. Funkce g zde s rostoucím n nabývá přibližně hodnoty π, překmit tedy činí asi 17 procent. Analogická situace platí i pro bod $x = -\pi + \frac{\pi}{n}$.

Jak je již jistě zřejmé, u nespojitých funkcí Gibbsův jev poměrně komplikuje použití částečných součtů k aproximaci funkce. Z tohoto důvodu se používá řada metod na jeho eliminaci. Uveďme alespoň metodu tzv. *σ-aproximace*, kdy se používají součty tvaru

$$\frac{a_0}{2} + \sum_{k=1}^{n-1} \operatorname{sinc}\left(\frac{k\pi}{n}\right) \left[a_k \cos(kx) + b_k \sin(kx)\right],$$

kde funkce $\operatorname{sinc} x$ je zadána předpisem

$$\operatorname{sinc} x = \begin{cases} \frac{\sin x}{x} & \text{pro } x \neq 0 \\ 1 & \text{pro } x = 0 \end{cases}.$$

K této metodě ještě vrátíme v části s řešenými příklady.

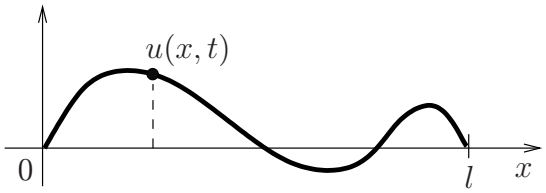
4.3 Použití Fourierových řad

Fourierovy řady mají mnoho aplikací v matematice či fyzice. Namátkou jmenujme například řešení parciálních diferenciálních rovnic, vyšetřování kmitavého pohybu, šíření vlnění, intenzity a napětí střídavého proudu atd.

V této části si předvedeme využití Fourierových řad při vyšetřování netlumeného harmonického kmitání struny upevněné na obou koncích. Vyjdeme z tzv. *vlnové rovnice*, pomocí které se popisuje například šíření zvuku, šíření elektromagnetických signálů a řada dalších jevů. V jednorozměrném případě má tato rovnice tvar

$$u_{tt} = a^2 u_{xx} \tag{4.6}$$

kde a je konstanta (pro danou strunu a sílu napínající strunu).



Obr. 4.5: Tvar struny v čase t.

Označme $u(x,t)$ výchylku bodu struny o úsečce x v čase t, viz. Obr. 4.5. Nechť v době $t = 0$ je struna vychýlena z rovnovážné polohy, takže má tvar křivky $\varphi(x)$. Působením síly napínající strunu se struna rozkmitá. Pro jednoduchost předpokládejme, že v čase $t = 0$

mají všechny body struny rychlost rovnu nule. Z matematického hlediska se úloha redukuje na nalezení funkce $u(x,t)$, která vyhovuje rovnici (4.6) a splňuje počáteční podmínky

$$u(x,0) = \varphi(x) \qquad u_t(x,0) = 0. \tag{4.7}$$

Tyto podmínky vyjadřují, že v době $t = 0$ má struna daný tvar a počáteční rychlost bodů struny je rovna nule. Dále musí být splněny hraniční podmínky

$$u(0,t) = 0 \qquad u(l,t) = 0, \tag{4.8}$$

které vyjadřují, že konce struny zůstávají v klidu.

Hledejme partikulární řešení úlohy (4.6), (4.7), (4.8) ve tvaru součinu dvou funkcí $X(x)$ a $T(t)$. Přímým dosazením se snadno přesvědčíme, že funkce

$$u_k(x,t) = C_k \cos\left(\frac{ak\pi}{l}\,t\right) \sin\left(\frac{k\pi}{l}\,x\right)$$

splňuje rovnici (4.6) pro libovolné C_k. Funkce také splňuje obě hraniční podmínky a druhou počáteční podmínku $u_t(x,0) = 0$. Aby byla splněna i první počáteční podmínka, položme $k = 1, 2, 3, \ldots$ a sestavme nekonečnou řadu z příslušných partikulárních řešení:

$$u(x,t) = \sum_{k=1}^{\infty} u_k(x,t) = \sum_{k=1}^{\infty} C_k \cos\left(\frac{ak\pi}{l}\,t\right) \sin\left(\frac{k\pi}{l}\,x\right) \tag{4.9}$$

Je zřejmé, že tato řada také vyhovuje rovnici (4.6). Abychom určily koeficienty C_k, položme $t = 0$. Získáváme

$$u(x,0) = \varphi(x) = \sum_{k=1}^{\infty} C_k \sin\left(\frac{k\pi}{l}\,x\right).$$

C_k jsou tedy Fourierovy koeficienty patřící k trigonometrické řadě, která je rozvojem funkce $\varphi(x)$ v sinovou řadu na intervalu $[0,l]$. Z části 1.2. již víme, že se jedná o Fourierovu řadu lichého rozšíření funkce $\varphi(x)$ na interval $[-l,l]$, tedy

$$C_k = \frac{2}{l} \int_0^l \sin\left(\frac{k\pi x}{l}\right) \varphi(x)\,\mathrm{d}x.$$

Také k této úloze se ještě vrátíme v části s řešenými příklady, kde ji budeme ilustrovat na konkrétním tvaru funkce $\varphi(x)$.

4.4 Modul FourierTrigSeries

Maple samotný nenabízí procedury pro počítání Fourierových řad a manipulaci s nimi. Z tohoto důvodu vzniklo několik externích procedur či rovnou celých knihoven s cílem automatizovat provádění výpočtů Fourierových řad. Tyto knihovny jsou často dostupné

prostřednictvím stránek Maple Application Center,[3] což je databáze zápisníků, knihoven, návodů a dalších zdrojů souvisejících s programem Maple.

Jmenujme například knihovnu `FourierSeries`, jejímž autorem je Amir Hussein Khanshan [32], stejnojmennou knihovnu Wilhelma Wernera [31], či autorův vlastní modul `Fourier` [18]. Všechny tyto knihovny poskytují nové procedury pro počítání Fourierových řad, částečných součtů, kreslení grafů a také provádění některých dalších výpočtů. Neřeší však manipulaci s řadami samotnými.

Zmíněné tři knihovny blíže popsal Robert J. Lopez v krátkém seriálu [30] věnovaném výpočtům Fourierových řad v programu Maple, publikovaném též prostřednictvím Maple Application Center.

V této části se blíže seznámíme s novou knihovnou `FourierTrigSeries`, která takřka plně pokrývá funkce zmíněných tří knihoven. Překonává je však v možnostech manipulace s Fourierovými řadami.

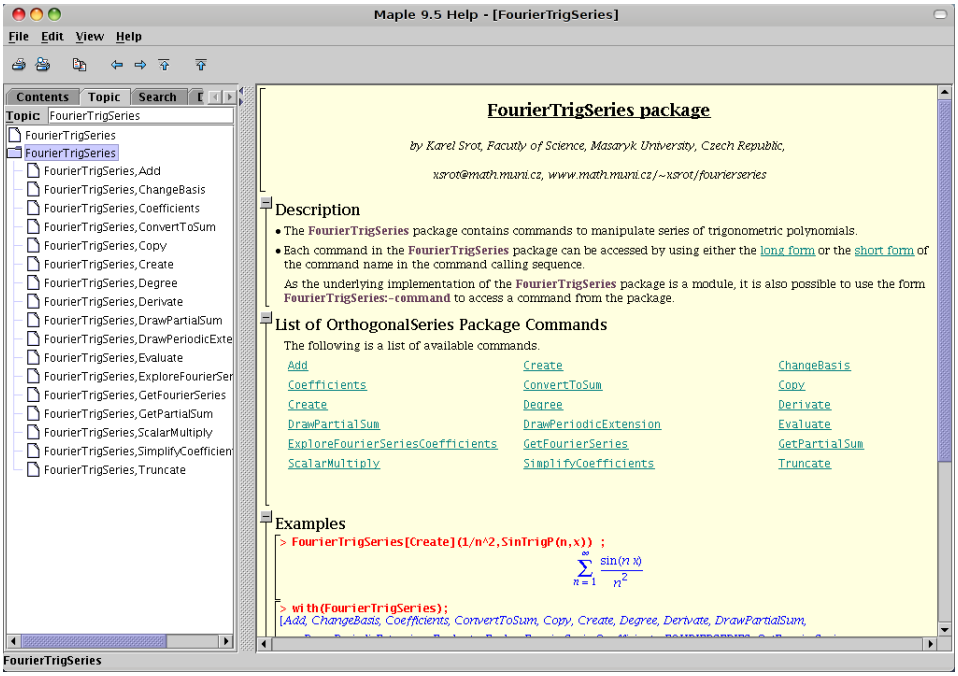
Vzorem pro tuto knihovnu byla knihovna `OrthogonalSeries`, jež je standardní součástí programu Maple a která slouží k provádění výpočtů s nekonečnými řadami ortogonálních polynomů.

Knihovna `FourierTrigSeries` zavádí nové datové typy pro reprezentaci trigonometrické řady a obsahuje řadu procedur, které se s těmito datovými typy operují. Většina procedur je blízká procedurám ze zmíněné knihovny `OrthogonalSeries`, často s nimi sdílí název, funkčnost i podobný způsob použití.

Jedná se o následující procedury:

- Add
- Coefficients
- ConvertToSum
- Copy
- Create
- Degree
- Derivate
- Evaluate
- ChangeBasis
- ScalarMultiply
- SimplifyCoefficients
- Truncate

[3] http://www.maplesoft.com/applications/index.aspx



Obr. 4.6: Nápověda ke knihovně FourierTrigSeries.

Dále je v knihovně `FourierTrigSeries` několik procedur, které svůj ekvivalent v knihovně OrthogonalSeries nemají, ale jejichž funkce úzce souvisí s tématem Fourierových řad.

- DrawPeriodicExtension
- DrawPartialSum
- ExploreFourierSeriesCoefficients
- GetFourierSeries
- GetPartialSum

Pokud je knihovna korektně nainstalována, je možné ji načíst příkazem

```
> with(FourierTrigSeries);
  Add, ChangeBasis, Coefficients, ConvertToSum, Copy, Create, Degree,
  Derivate, DrawPartialSum, DrawPeriodicExtension, Evaluate,
  ExploreFourierSeriesCoefficients, FOURIERSERIES, GetFourierSeries,
  GetPartialSum, SERIESORTHOGONALSYSTEM, ScalarMultiply,
  SimplifyCoefficients, Truncate
```

Dostupná je také nápověda s bližším popisem všech procedur, která je plně integrována do prostředí nápovědy programu Maple (viz obrázek 4.6).

V následujícím textu si jednotlivé procedury blíže představíme a předvedeme způsob jejich použití. Mějme prosím na paměti, že dále uvedené příklady jsou ilustrativní. Jako parametry procedur jsou záměrně voleny velmi jednoduché výrazy, na nichž je pak funkčnost

procedur samotných dobře patrná. Z tohoto důvodu není naprostá většina trigonometrických řad, vystupujících v této kapitole, konvergentní.

Reprezentace trigonometrické řady v knihovně `FourierTrigSeries`

Nekonečná trigonometrická řada je reprezentována novým datovým typem `FOURIERSERIES`. Ten obsahuje charakteristiky této řady, například hodnoty koeficientů trigonometrické řady a použitý ortogonální systém.

Samotný ortogonální systém je reprezentován novým typem `SERIESORTHOGONALSYSTEM`, který představují následující čtyři definované konstanty: `CosSinTrigP` v případě obecné trigonometrické řady,[4] `CosTrigP` respektive `SinTrigP` v případě kosinové respektive sinové řady a konečně `ExpTrigP2` v případě řady v komplexním (nebo též exponenciálním) tvaru.

Nebude-li řečeno jinak, budeme v následujícím popisu procedur předpokládat, že trigonometrická řada je reprezentována právě typem `FOURIERSERIES`.

Procedura `Create`

Procedura `Create` slouží k vytvoření trigonometrické řady, která je reprezentovaná typem `FOURIERSERIES`. V parametrech procedury je nutné ve speciální formě zadat základní charakteristiky trigonometrické řady, konkrétně hodnoty koeficientů, zvolený ortogonální systém a samozřejmě také proměnné reprezentující index a neznámou. Proceduře `Create` je možné předávat parametry několika různými způsoby, které se mimo jiné liší i v závislosti na použitém ortogonálním systému. Jednotlivé způsoby jsou blíže popsány v nápovědě. Konkrétní použití procedury `Create` pak může vypadat následovně:

```
> S:=Create({[1,[1,-1],[1/2,-1/2]], 'general'=[n,-n]}, CosSinTrigP(n,x));
```

$$S := 1 + \cos(x) - \sin(x) + \frac{1}{2}\cos(2x) - \frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{\infty}(n\cos(nx) - n\sin(nx))\right)$$

V tomto případě jsme uvedli konkrétní hodnoty koeficientů a_0, a_1, b_1, a_2 a b_2 a dále zadali obecný předpis pro hodnoty koeficientů zbývajících. Koeficienty jsou zadávany po dvojicích, jedna dvojice se vztahuje k příslušným kosinovým a sinovým členům. Výjimku tvoří absolutní člen řady, jež je uveden jako první.

Ačkoliv to z předchozího výstupu není patrné,[5] získaná řada není standardní řadou vzniklou pomocí procedury `Sum`. Její vnitřní struktura je od této řady odlišná, zobrazit ji můžeme pomocí příkazu `lprint`.

```
> lprint(S);
  `FourierTrigSeries:-FOURIERSERIES`(S)
```

[4] Pod tímto pojmem se rozumí trigonometrická řada, ve které vystupují sinové i kosinové členy.

[5] Z důvodu větší přehlednosti a lepší čitelnosti je před výstupem vizuální reprezentace typu `FOURIERSERIES` konvertována procedurou `ConvertToSum` na „běžnou“ nekonečnou řadu.

```
> print(op(1,S));
  ARRAY([0 .. 1], [(0) = TABLE(sparse, ["bounds" = [3, infinity],
  "table" = TABLE(sparse, [0 = [1, 0], 1 = [1, -1], 2 = [1/2, (-1)/2]]),
  "dim" = 1, "general" = [n, -n]]), (1) = TABLE(["period" = 2*Pi,
  "genre" = CosSinTrigP, "index" = n, "variable" = x])])
```

Ověřit typ funkční řady je také možné pomocí standardní procedury `type`.

```
> type(S, FOURIERSERIES);
```

$$true$$

V případě sinové a kosinové řady je použití procedury lehce odlišné, jelikož koeficienty trigonometrické řady již nezadáváme po dvojicích.

```
> Create({[1,2,3], n}, CosTrigP(n,x));
```

$$1 + \cos(x) + 2\cos(2x) + 3\cos(3x) + \left(\sum_{n=3}^{\infty} n\cos(nx)\right)$$

Pokud si přejeme explicitně specifikovat pouze některé koeficienty, není nezbytné uvádět všechny koeficienty předchozí.

```
> Create({[4=[1/2,-1/2], 8=[2,-2]], 'general'=[0,0]}, CosSinTrigP(n,x));
```

$$\frac{1}{2}\cos(4x) - \frac{1}{2}\sin(4x) + 2\cos(8x) - 2\sin(8x)$$

Předchozí příklad také ilustroval způsob, jak definovat konečný rozvoj. Druhou možností je explicitní omezení počtu členů funkční řady.

```
> Create({k, k=1..5}, SinTrigP(k,t));
```

$$\sum_{k=1}^{5} n\sin(kt)$$

Pomocí volitelného parametru je také možné explicitně zadat periodu rozvoje (implicitně je perioda rovna 2π). To nám umožní omezit se pouze na rozvoje, jejichž členy jsou tvaru

$$a_n \cos\left(\frac{2\pi}{p}nx\right) + b_n \sin\left(\frac{2\pi}{p}nx\right),$$

kde $p \in \mathbb{R}^+$ je uvažovaná perioda.

V následujícím případě je zadána perioda rovna π, ve výsledném rozvoji tak budou vystupovat pouze trigonometrické polynomy sudého stupně.

```
> Create({[1,2,3], n}, SinTrigP(n,x), Pi);
```

$$\sin(2x) + 2\sin(4x) + 3\sin(6x) + \left(\sum_{n=4}^{\infty} n\sin(2nx)\right)$$

Při výpočtech samotných Fourierových řad nebudeme proceduru `Create` používat přímo. Využijeme ji ale například při řešení diferenciálních rovnic metodou Fourierových řad.

Procedury k manipulaci s typem FOURIERSERIES

Procedura Add

Procedura `Add` vrací součet dvou trigonometrických řad. Tyto trigonometrické řady musí mít stejnou periodu a musí být kompatibilních typů. Zatímco řady s ortogonálními systémy `CosSinTrigP`, `CosTrigP` a `CosTrigP` lze sčítat mezi sebou libovolně, řadu s ortogonálním systémem `ExpTrigP2` je možné sečíst pouze s řadou s týmž ortogonálním systémem.[6]

Kromě dvou povinných parametrů, kterými jsou sčítané trigonometrické řady, je možné specifikovat až dva další nepovinné parametry (implicitně jsou rovny jedné). Výstupem procedury je pak lineární kombinace obou řad přičemž v roli koeficientů této lineární kombinace vystupují právě tyto nepovinné parametry.

```
> S:=Create({[1],n}, CosTrigP(n,x));
```

$$S := 1 + \left(\sum_{n=1}^{\infty} n \cos(nx) \right)$$

```
> S2:=Create({n}, SinTrigP(n,x));
```

$$S2 := \sum_{n=1}^{\infty} n \sin(nx)$$

```
> Add(S, S2);
```

$$1 + \left(\sum_{n=1}^{\infty} (n \cos(nx) + n \sin(nx)) \right)$$

```
> Add(S, S2, a, b);
```

$$a + \left(\sum_{n=1}^{\infty} (an \cos(nx) + bn \sin(nx)) \right)$$

Procedura Copy

Procedura `Copy` vytváří kopii zadané trigonometrické řady.[7] Ačkoliv momentálně není možné procedurami z knihovny `FourierTrigSeries` přímo měnit charakteristiky trigonometrické řady,[8] situace se může změnit v některé z následujících verzí.

```
> S:=Create({[1,[1,-1],[1/2,-1/2]], 'general'=[n,-n]}, CosSinTrigP(n,x));
```

$$S := 1 + \cos(x) - \sin(x) + \frac{1}{2}\cos(2x) - \frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{\infty} (n \cos(nx) - n \sin(nx)) \right)$$

[6] Samozřejmě je možné řady nekompatibilních typů sečíst po konverzi jedné z nich procedurou `ChangeBasis`.

[7] V případě přiřazení `T:=S;` ukazují obě proměnné na stejnou interní strukturu definovanou v programu Maple. Proto je procedura `Copy` nezbytná pro vytvoření „nezávislého“ duplikátu původní řady.

[8] Stejně je tomu i v případě knihovny `OrthogonalSeries`

```
> S2:=Copy(S);
```

$$S2 := 1 + \cos(x) - \sin(x) + \frac{1}{2}\cos(2x) - \frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{\infty}(n\cos(nx) - n\sin(nx))\right)$$

Procedura `Derivate`

Procedura `Derivate` slouží k derivování trigonometrické řady podle zadané neznámé.

```
> S:=Create({[t], t*n}, CosTrigP(n,x));
```

$$S := t + \left(\sum_{n=1}^{\infty} tn\cos(nx)\right)$$

```
> Derivate(S, x);
```

$$\sum_{n=1}^{\infty}(-n^2 t\sin(nx))$$

```
> Derivate(S, t);
```

$$1 + \left(\sum_{n=1}^{\infty} n\cos(nx)\right)$$

Procedura `ChangeBasis`

Procedura `ChangeBasis` slouží k převodu trigonometrické řady obsahující sinové a kosinové členy na řadu v komplexním (exponenciálním) tvaru (a naopak). Konkrétně se jedná o převod mezi definovanými systémy `CosSinTrigP` (respektive `CosTrigP` a `SinTrigP`) a systémem `ExpTrigP2`.

```
> S:=Create({[1,[1,-1],[1/2,-1/2]], 'general'=[n,-n]}, CosSinTrigP(n,x));
```

$$S := 1 + \cos(x) - \sin(x) + \frac{1}{2}\cos(2x) - \frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{\infty}(n\cos(nx) - n\sin(nx))\right)$$

```
> S2:=ChangeBasis(S, ExpTrigP2(n,x));
```

$$S2 := 1 + \left(\frac{1}{2} + \frac{1}{2}I\right)\mathrm{e}^{(Ix)} + \left(\frac{1}{2} - \frac{1}{2}I\right)\mathrm{e}^{(-Ix)} + \left(\frac{1}{4} + \frac{1}{4}I\right)\mathrm{e}^{(2Ix)} + \left(\frac{1}{4} - \frac{1}{4}I\right)\mathrm{e}^{(-2Ix)} + \\ + \left(\sum_{n=3}^{\infty}\left(\left(\frac{1}{2}n + \frac{1}{2}In\right)\mathrm{e}^{(Inx)} + \left(\frac{1}{2}n - \frac{1}{2}In\right)\mathrm{e}^{(-Inx)}\right)\right)$$

```
> ChangeBasis(S2, SinTrigP(n,x));
```

'Cannot convert to SinTrigP, converting to CosSinTrigP'

$$1 + \cos(x) - \sin(x) + \frac{1}{2}\cos(2x) - \frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{\infty}(n\cos(nx) - n\sin(nx))\right)$$

V posledním případě nebylo možné řadu s ortogonálním systémem `ExpTrigP2` převést na požadovaný ortogonální systém `SinTrigP`, proto byla automaticky převedena na ortogonální systém `CosSinTrigP`.

Procedura `ScalarMultiply`

Pomocí procedury `ScalarMultiply` je možné vynásobit koeficienty trigonometrické řady konstantou či algebraickým výrazem. Tento však nesmí záviset na neznámé a indexu, které v této trigonometrické řadě vystupují.

```
> S:=Create({[1,[1,-1],[1/2,-1/2]], 'general'=[n,-n]}, CosSinTrigP(n,x));
```

$$S := 1 + \cos(x) - \sin(x) + \frac{1}{2}\cos(2x) - \frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{\infty}(n\cos(nx) - n\sin(nx))\right)$$

```
> ScalarMultiply(S, 1-alpha);
```

$$1 - \alpha + (1-\alpha)\cos(x) + (-1+\alpha)\sin(x) + \left(\frac{1}{2} - \frac{1}{2}\alpha\right)\cos(2x) +$$
$$+ \left(-\frac{1}{2} + \frac{1}{2}\alpha\right)\frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{\infty}((1-\alpha)n\cos(nx) - (1-\alpha)n\sin(nx))\right)$$

Procedura `SimplifyCoefficients`

Tato procedura slouží k úpravám koeficientů trigonometrické řady. Jejím prostřednictvím je možné na koeficienty trigonometrické řady například aplikovat proceduru `simplify` a tak tyto koeficienty zjednodušit.

```
> S:=Create(-1/(n-1), SinTrigP(n,x));
```

$$S := \sum_{n=1}^{\infty}\left(-\frac{\sin(nx)}{n-1}\right)$$

```
> S2:=Create(n/(n-1), SinTrigP(n,x));
```

$$S2 := \sum_{n=1}^{\infty}\left(\frac{n\sin(nx)}{n-1}\right)$$

```
> Add(S, S2);
```

$$\sum_{n=1}^{\infty}\left(\frac{n}{n-1} - \frac{1}{n-1}\right)\sin(nx)$$

```
> SimplifyCoefficients(%, simplify);
```

$$\sum_{n=1}^{\infty}\sin(nx)$$

Procedura `Truncate`

Výstupem procedury `Truncate` je částečný součet zadané trigonometrické řady. Tento částečný součet je stále reprezentován typem `FOURIERSERIES`.

```
> S:=Create({[1,[1,-1],[1/2,-1/2]], 'general'=[n,-n]}, CosSinTrigP(n,x));
```

$$S := 1 + \cos(x) - \sin(x) + \frac{1}{2}\cos(2x) - \frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{\infty}(n\cos(nx) - n\sin(nx))\right)$$

```
> S2:=Truncate(S, 5);
```

$$S2 := 1 + \cos(x) - \sin(x) + \frac{1}{2}\cos(2x) - \frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{5}(n\cos(nx) - n\sin(nx))\right)$$

```
> lprint(S2);
  `FourierTrigSeries:-FOURIERSERIES`(S2)
```

Další procedury pracující s typem `FOURIERSERIES`

Procedura `Coefficients`

Tato procedura slouží k výpisu hodnot koeficientů nekonečné řady. V případě, že je tato řada jedinným parametrem procedury, je výsledkem obecný předpis pro hodnoty koeficientů.

```
> S:=Create({[1,[1,-1],[1/2,-1/2]], 'general'=[n,-n]}, CosSinTrigP(n,x));
```

$$S := 1 + \cos(x) - \sin(x) + \frac{1}{2}\cos(2x) - \frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{\infty}(n\cos(nx) - n\sin(nx))\right)$$

```
> Coefficients(S);
```

$$[n, -n]$$

Pomocí druhého parametru se odkazujeme na konkrétní člen řady.

```
> Coefficients(S, 0);
```

$$1$$

```
> Coefficients(S, 2);
```

$$\left[\frac{1}{2}, -\frac{1}{2}\right]$$

```
> Coefficients(S, 5);
```

$$[5, -5]$$

Procedura `Degree`

Výstupem procedury `Degree` je stupeň trigonometrické řady. V případě konečného rozvoje je roven stupni tohoto trigonometrického polynomu. V případě nekonečné řady je její stupeň roven nekonečnu.

```
> S:=Create({[1,[1,-1],[1/2,-1/2]], 'general'=[n,-n]}, CosSinTrigP(n,x));
```

$$S := 1 + \cos(x) - \sin(x) + \frac{1}{2}\cos(2x) - \frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{\infty}(n\cos(nx) - n\sin(nx))\right)$$

```
> Degree(S);
```

$$\infty$$

```
> S2:=Truncate(S, 5);
```

$$S2 := 1 + \cos(x) - \sin(x) + \frac{1}{2}\cos(2x) - \frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{5}(n\cos(nx) - n\sin(nx))\right)$$

```
> Degree(S2);
```

$$5$$

Procedura `ConvertToSum`

Procedura `ConvertToSum` slouží ke konverzi funkční řady reprezentované pomocí typu `FOURIERSERIES` na „obyčejnou“ řadu vzniklou použitím procedury `Sum`. Jedinným parametrem procedury je řada typu `FOURIERSERIES`.

```
> S:=Create({[1,[1,-1],[1/2,-1/2]], 'general'=[n,-n]}, CosSinTrigP(n,x));
```

$$S := 1 + \cos(x) - \sin(x) + \frac{1}{2}\cos(2x) - \frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{\infty}(n\cos(nx) - n\sin(nx))\right)$$

```
> lprint(S);
  `FourierTrigSeries:-FOURIERSERIES`(S)
> S2:=ConvertToSum(S);
```

$$S2 := 1 + \cos(x) - \sin(x) + \frac{1}{2}\cos(2x) - \frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{\infty}(n\cos(nx) - n\sin(nx))\right)$$

```
> lprint(S2);
  1+cos(x)-sin(x)+1/2*cos(2*x)-1/2*sin(2*x)+(Sum(n*cos(n*x)-n*sin(n*x), n = 3
  .. infinity))
```

Takto reprezentovanou řadu využijeme v případech, kdy je reprezentace řady pomocí datového typu `FourierTrigSeries` pro další výpočty nevhodná.

Procedura `Evaluate`

Pomocí procedury `Evaluate` můžeme do trigonometrické řady za neznámou dosadit konkrétní výraz.

```
> S:=Create({n}, CosTrigP(n,x));
```

$$S := \sum_{n=1}^{\infty} n\cos(nx)$$

```
> Evaluate(S, x=Pi);
```

$$S := \sum_{n=1}^{\infty} n\cos(n\pi)$$

Pomoci procedury `Evaluate` s parametrem `trunc` lze také získat částečný součet trigonometrické řady.

```
> Evaluate(S, trunc=5);
```

$$\cos(x) + 2\cos(2x) + 3\cos(3x) + 4\cos(4x) + 5\cos(5x)$$

Předchozí dvě možnosti lze také vzájemně kombinovat.

```
> Evaluate(s, x=Pi/4, trunc=5);
```

$$-4 - \frac{7}{2}\sqrt{2}$$

Procedura `GetPartialSum`

Podobně jako u procedury `Truncate`, také výstupem procedury `GetPartialSum` je částečný součet zadané trigonometrické řady. V jejím případě je ale tento částečný součet automaticky konvertován procedurou `ConvertToSum` na běžně používanou reprezentaci, získanou pomocí procedury `Sum`.

```
> S:=Create({[1,[1,-1],[1/2,-1/2]], 'general'=[n,-n]}, CosSinTrigP(n,x));
```

$$S := 1 + \cos(x) - \sin(x) + \frac{1}{2}\cos(2x) - \frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{\infty}(n\cos(nx) - n\sin(nx))\right)$$

```
> S2:=GetPartialSum(S, 5);
```

$$S2 := 1 + \cos(x) - \sin(x) + \frac{1}{2}\cos(2x) - \frac{1}{2}\sin(2x) + \left(\sum_{n=3}^{5}(n\cos(nx) - n\sin(nx))\right)$$

```
> lprint(S2);
1+cos(x)-sin(x)+1/2*cos(2*x)-1/2*sin(2*x)+(Sum(n*cos(n*x)-n*sin(n*x), n = 3
.. 5))
```

Procedury pro výpočty Fourierových řad

Procedura `GetFourierSeries`

Procedura `GetFourierSeries` počítá Fourierovu řadu zadané funkce na daném intervalu. Výsledná Fourierova řada je reprezentována typem `FOURIERSERIES`.

Prvním parametrem této procedury musí být reálná funkce, druhým pak proměnná a interval, na kterém Fourierovu řadu počítáme.

```
> GetFourierSeries(cos(x)^4, x=-Pi..Pi);
```

$$\frac{3}{8}+\frac{1}{2}\cos(2x)+\frac{1}{8}\cos(4x)$$

```
> GetFourierSeries(x^2, x=-Pi..Pi);
```

$$\frac{1}{3}\pi^2+\left(\sum_{n=1}^{\infty}\frac{4(-1)^n\cos(nx)}{n^2}\right)$$

Tuto řadu můžeme pomocí procedury `ChangeBasis` převést do komplexního (exponenciálního) tvaru. Téhož ale docílíme přímo použitím volby `exp`.

```
> GetFourierSeries(x^2, x=-Pi..Pi, exp);
```

$$\frac{1}{3}\pi^2+\left(\sum_{n=1}^{\infty}\left(\frac{2(-1)^n\mathrm{e}^{(Inx)}}{n^2}+\frac{2(-1)^n\mathrm{e}^{(-Inx)}}{n^2}\right)\right)$$

Procedurou `GetFourierSeries` lze počítat také sinový respektive kosinový rozvoj dané funkce (tedy Fourierovu řadu lichého respektive sudého periodického rozšíření této funkce), k tomu slouží volby `odd` a `even`. V tomto případě musí být jeden z krajních bodů intervalu roven nule.

```
> GetFourierSeries(x,x=0..Pi, odd);
```

$$\sum_{n=1}^{\infty}\left(-\frac{2(-1)^n\sin(nx)}{n}\right)$$

```
> GetFourierSeries(x,x=0..Pi, even);
```

$$\frac{1}{2}\pi+\left(\sum_{n=1}^{\infty}\frac{2((-1)^n-1)\cos(nx)}{\pi n^2}\right)$$

Také volby `odd` a `even` můžeme kombinovat s volbou `exp`.

```
> GetFourierSeries(x,x=0..Pi, even, exp);
```

$$\frac{1}{2}\pi+\left(\sum_{n=1}^{\infty}\left(\frac{((-1)^n-1)\mathrm{e}^{(Inx)}}{\pi n^2}+\frac{((-1)^n-1)\mathrm{e}^{(-Inx)}}{\pi n^2}\right)\right)$$

Procedura `ExploreFourierSeriesCoefficients`

Výstupem procedury `GetFourierSeries` je již celá Fourierova řada. Naopak proceduru `ExploreFourierSeriesCoefficients` využijeme v případě, kdy chceme znát pouze hodnoty koeficientů Fourierovy řady. Kromě samotných koeficientů je dále výstupem procedury také formální tvar Fourierovy řady.

```
> ExploreFourierSeriesCoefficients(signum(sin(2*x)), x=-Pi..Pi);
```

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

$$a_n = 0$$

$$b_n = \frac{2\left((-1)^n + 1 - 2\cos\left(\frac{1}{2}n\pi\right)\right)}{\pi n}$$

Tyto informaci lze sice získat také následujícími příkazy,

```
> GetFourierSeries(signum(sin(2*x)), x=-Pi..Pi);
```

$$\sum_{n=1}^{\infty}\frac{2\left((-1)^n + 1 - 2\cos\left(\frac{1}{2}n\pi\right)\right)\sin(nx)}{\pi n}$$

```
> Coefficients(%);
```

$$\left[0, \frac{2\left((-1)^n + 1 - 2\cos\left(\frac{1}{2}n\pi\right)\right)}{\pi n}\right]$$

ale procedura `ExploreFourierSeriesCoefficients` umožňuje tyto koeficienty prozkoumat detailněji a to pomocí třetího (nepovinného) parametru, jehož hodnota určuje požadovanou „úroveň detailu“ (implicitně je roven 1).

```
> ExploreFourierSeriesCoefficients(signum(sin(2*x)), x=-Pi..Pi, 2);
```

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

$$a_n = 0$$

$$b_n = \begin{cases} -\frac{2(-1+(-1)^k)}{\pi k} & n = 2k \\ 0 & n = 2k+1 \end{cases}$$

```
> ExploreFourierSeriesCoefficients(signum(sin(2*x)), x=-Pi..Pi, 4);
```

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

$$a_n = 0$$

$$b_n = \begin{cases} 0 & n = 4k \\ 0 & n = 4k+1 \\ \frac{4}{\pi(2k+1)} & n = 4k+2 \\ 0 & n = 4k+3 \end{cases}$$

Pomocí volby `exp` lze opět pracovat s řadou v komplexním tvaru.

```
> ExploreFourierSeriesCoefficients(signum(sin(2*x)), x=-Pi..Pi, 4, exp);
```

$$c_0 + \left(\sum_{n=1}^{\infty} \left(c_n \mathrm{e}^{(Inx)} + c_{-n} \mathrm{e}^{(-Inx)} \right) \right)$$

$$c_0 = 0$$

$$c_n = \begin{cases} 0 & n = 4k \\ 0 & n = 4k+1 \\ -\frac{2I}{\pi(2k+1)} & n = 4k+2 \\ 0 & n = 4k+3 \end{cases}$$

$$c_{-n} = \begin{cases} 0 & n = 4k \\ 0 & n = 4k+1 \\ \frac{2I}{\pi(2k+1)} & n = 4k+2 \\ 0 & n = 4k+3 \end{cases}$$

V případě konečného rozvoje jsou explicitně uvedeny hodnoty všech nenulových koeficientů.

```
> ExploreFourierSeriesCoefficients(cos(x)^4, x=-Pi..Pi);
```

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

$$a_n = \begin{cases} \frac{3}{4} & n = 0 \\ \frac{1}{2} & n = 2 \\ \frac{1}{8} & n = 4 \\ 0 & otherwise \end{cases}$$

$$b_n = 0$$

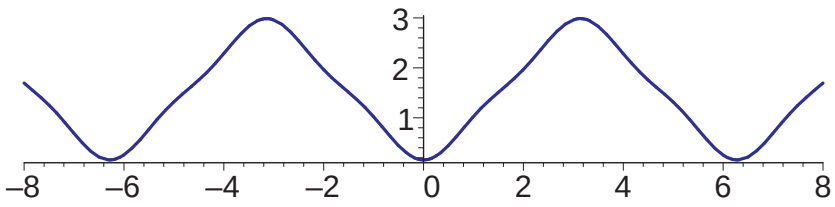
Procedury s grafickým výstupem

Procedura `DrawPartialSum`

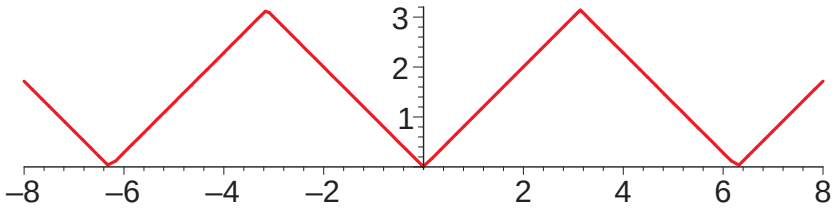
Procedura `DrawPartialSum` usnadňuje kreslení grafů částečných součtů Fourierových řad reprezentovaných typem `FOURIERSERIES`. Je ekvivalentní postupnému použití procedur `GetPartialSum` a `plot`.[9] Očekává dva povinné parametry. Prvním je Fourierova řada a druhým index částečného součtu. Dále lze využít širokou paletu voleb, jež ovlivňují podobu výsledného grafu a které jsou shodné s volbami používanými v proceduře `plot`.[10]

[9] Nebo také `Truncate`, `ConvertToSum` a `plot`.

[10] Popis těchto voleb je dostupný prostřednictvím nápovědy, například zadáním příkazu `?plot/options`.



Obr. 4.7: Výstup procedury `DrawPartialSum`, graf částečného součtu s_3.



Obr. 4.8: Sudé periodické rozšíření funkce $f(x) = x$, $x \in [0, \pi]$.

```
> S:=GetFourierSeries(x,x=0..Pi,even):
> DrawPartialSum(S, 3, -8..8, scaling=constrained, color=blue);
```

Výsledný graf je na obrázku 4.7.

Procedura `DrawPeriodicExtension`

Procedura `DrawPeriodicExtension` kreslí graf periodického rozšíření zadané funkce. Využití najde zejména při ilustraci konvergence Fourierovy řady prostřednictvím grafů a animací s částečnými součty.

Prvním parametrem je požadovaná funkce, druhým parametrem proměnná a základní interval a třetím parametrem horizontální rozsah výsledného grafu. Dále je možné uvést různé volby ovlivňující podobu výsledného grafu, podobně jako tomu bylo u procedury `DrawPartialSum`.

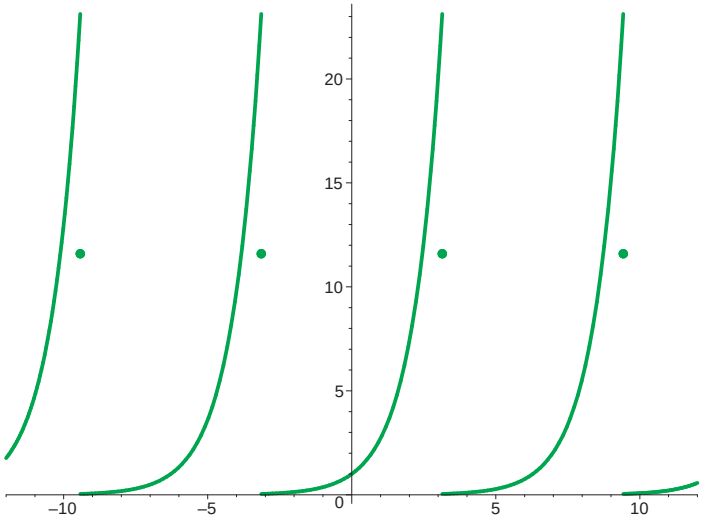
Navíc jsou k dispozici další tři volby. Volby `odd` a `even` slouží pro kreslení lichého respektive sudého periodického rozšíření.

```
> DrawPeriodicExtension(x, x=0..Pi, -8..8, even, scaling=constrained);
```

Volba `drawlimits` zvýrazní limity Fourierovy řady zadané funkce v bodech nespojitosti této funkce.

```
> DrawPeriodicExtension(exp(x), x=-Pi..Pi, -12..12, drawlimits,
  color=aquamarine);
```

Výstupy předchozích dvou příkazů jsou na obrázcích 4.8 a 4.9.



Obr. 4.9: 2π-periodické rozšíření funkce $f(x) = \mathrm{e}^x$.

Instalace knihovny `FourierTrigSeries`

Nejnovější verze[11] knihovny `FourierTrigSeries` lze získat na domovské stránce projektu.[12] Její funkčnost v programu Maple byla testována ve verzích 8, 9, 9.5, 10 a 11.

Následující postup instalace se týká Maplu 8, lze jej však realizovat i v jiných verzích.

1. Modul se skládá ze souborů `maple.lib`, `maple.ind` a `maple.hdb`.[13] Nejdříve je třeba vytvořit nový adresář a tyto soubory do tohoto adresáře zkopírovat. Předpokládejme, že soubory knihovny se nachází v adresáři `c:/devel/maple/fouriertrigseries`[14].

2. Jména adresářů, ve kterých se moduly hledají, jsou uloženy v proměnné `libname`. Pro použití modulu tedy stačí zadat následující příkaz.

   ```
   > libname:=libname, "c:/devel/maple/fouriertrigseries":
   ```

3. Abychom tento příkaz nemuseli zadávat při každém spuštění programu Maple, vytvoříme tzv. inicializační soubor, do kterého příkaz vložíme. Na platformě Windows se tento soubor jmenuje `maple.ini`[15] a Maple jej hledá v adresářích `"Maple 8/Lib"`[16],

[11] Aktuálně se jedná o verzi 0.41.

[12] `http://www.math.muni.cz/~xsrot/fourierseries`

[13] Soubor `maple.hdb` obsahuje stránky nápovědy ke knihovně `FourierTrigSeries`, k běhu samotných procedur není nezbytný.

[14] V Maplu lze místo zpětného lomítka použít v cestě lomítko běžné. Zpětné lomítko je speciální znak a bylo by tedy nutné jej psát zdvojeně, navíc se tak snižují rozdíly mezi platformami Unix popř. Linux a Windows.

[15] Pozor, nezaměňovat se souborem `maple8.ini` či souborem podobným v jiných verzích.

[16] `"Maple 8"` zde označuje adresář ve kterém je Maple nainstalován.

v pracovním adresáři, `"Maple 8/Users"` či adresáři s uživatelovým profilem. V Unixových systémech se inicializační soubor jmenuje `.mapleinit` a musíme jej umístit do svého domovského adresáře.

4. Po spuštění programu Maple by již mělo být možné knihovnu načíst příkazem

```
> with(FourierTrigSeries);
```

On-line výpočty Fourierových řad

Součástí webových stránek o knihovně `FourierTrigSeries` je i webová aplikace *Fourier trigonometric series calculator*. Aplikace demonstruje možnosti některých procedur knihovny `FourierTrigSeries` a s využitím platformy MapleNet umožňuje provádět výpočty Fourierových řad on-line pouze prostřednictvím internetového prohlížeče.

Rozhraní pro výpočty je složeno ze čtyř formulářů. První formulář slouží k zadání dat nezbytných pro všechny tři dostupné druhy výpočtů, konkrétně se jedná o předpis vyšetřované funkce a interval pro výpočet rozvoje. Nezbytné je při zadávání dat použít syntaxe programu Maple.

Prostřednictvím druhého formuláře se provádí výpočet Fourierovy řady zadané funkce, výsledná řada je vrácena v trigonometrickém a komplexním tvaru. Při výpočtu je tedy využita procedura `GetFourierSeries`. Tato situace je zachycena na obrázku 4.10.

Prostřednictvím třetího formuláře se počítají částečné součty Fourierovy řady (opět v trigonometrickém i komplexním tvaru). V tomto případě je nezbytné konkretizovat požadovaný částečný součet. Součástí výsledku je i graf získaného částečného součtu. Při výpočtu jsou využity procedury `GetFourierSeries` a `GetPartialSum`.

Poslední formulář umožňuje bližší pohled na hodnoty koeficientů Fourierovy řady ve smyslu, jak to umožňuje procedura `ExploreFourierSeriesCoefficients`.

4.5 Řešené příklady

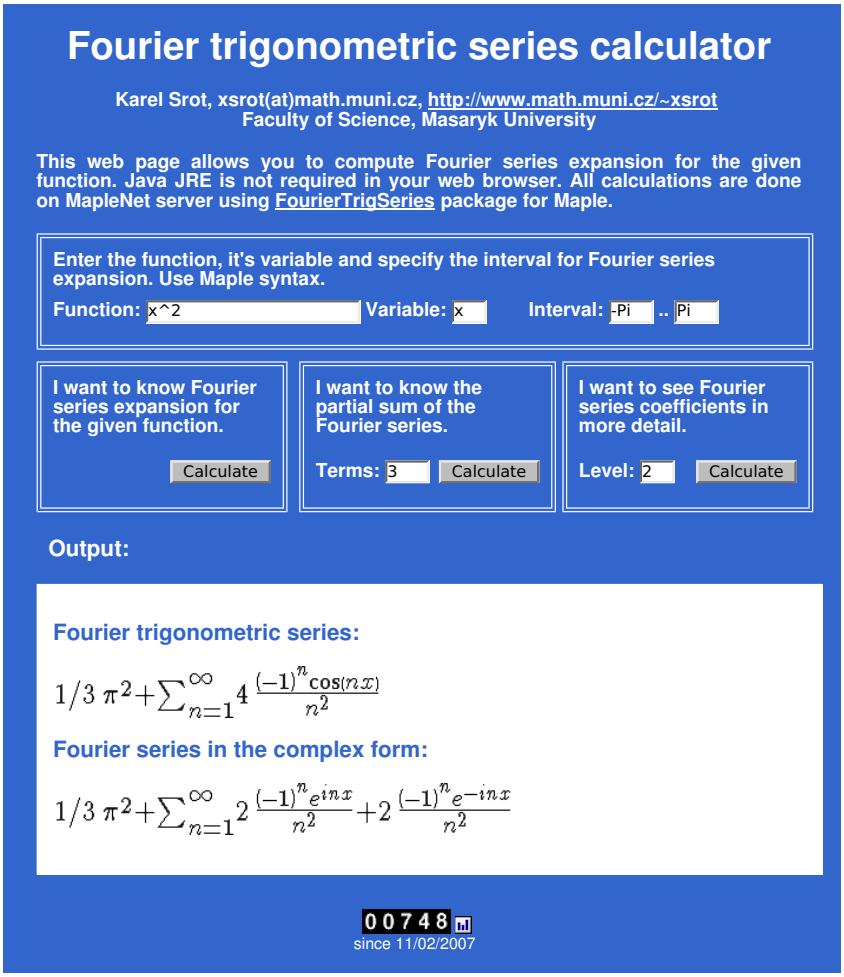
V této kapitole využijeme procedur modulu `Fourier` při řešení několika příkladů. Přesto je v některých případech nutné při výpočtu asistovat „ručně“.

Příklad 4.6. *Vypočítejte Fourierovu řadu funkce* $y = \frac{\pi}{2} - \frac{x}{2}$ *na intervalu* $[0, 2\pi]$ *a vykreslete grafy částečných součtů* s_1, s_2, s_4 *a* s_8.

Řešení. Pro výpočet Fourierovy řady použijeme proceduru `GetFourierSeries`. Výslednou řadu uložíme do proměnné a použijeme ji při kreslení grafu.

```
> f:=x->Pi/2-x/2:
> FS:=GetFourierSeries(f(x),x=0..2*Pi);
```

$$FS := \sum_{n=1}^{\infty} \frac{\sin(nx)}{n}$$



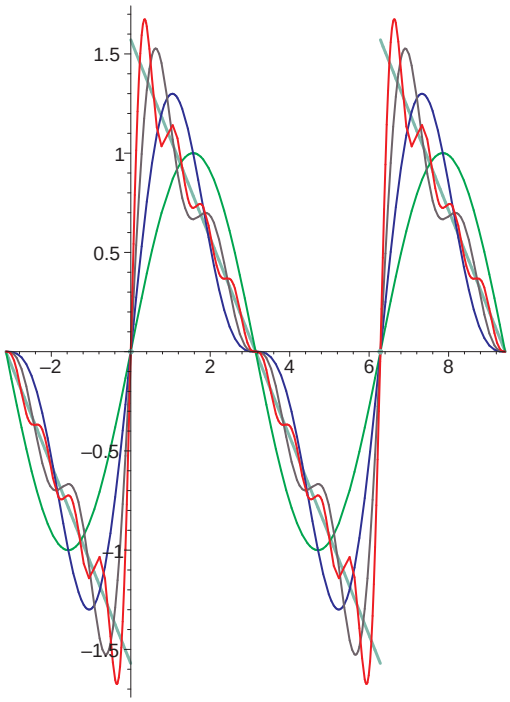
Obr. 4.10: Webové rozhraní aplikace Fourier trigonometric series calculator.

Pomocí grafu si znázorníme periodické rozšíření funkce f a vybrané částečné součty.

```
> gb:=DrawPeriodicExtension(f(x),x=0..2*Pi,-Pi..3*Pi,drawlimits,
  color=aquamarine,thickness=2):
> g1:=DrawPartialSum(FS, 1, -Pi..3*Pi, color=green):
> g2:=DrawPartialSum(FS, 2, -Pi..3*Pi, color=blue):
> g3:=DrawPartialSum(FS, 4, -Pi..3*Pi, color=violet):
> g4:=DrawPartialSum(FS, 8, -Pi..3*Pi, color=red):
> plots[display](gb, g1, g2, g3, g4);
```

Výsledný graf je na obrázku 4.11. □

Příklad 4.7. *Určete Fourierovu řadu funkce* $y = \cos(ax)$ *s parametrem* $a \in \mathbb{R} - \{0\}$ *na intervalu* $[-1, 1]$.



Obr. 4.11: Periodické rozšíření funkce $y = \frac{\pi}{2} - \frac{x}{2}$ a vybrané částečné součty.

Řešení.

```
> f:=x->cos(a*x):
> GetFourierSeries(f,x=-1..1);
```

$$\frac{\sin a}{a} + \sum_{n=1}^{\infty} \frac{2a(-1)^n \sin a \cos(\pi n x)}{a^2 - \pi^2 n^2}$$

□

Příklad 4.8. *Vypočítejte Fourierovu řadu funkce zadané předpisem*

$$f(x) = \begin{cases} a & pro\ x < 0 \\ b & pro\ x \geq 0 \end{cases}$$

na intervalu $[-\pi, \pi]$*, kde* a*,* b *jsou pevně zvolené reálné parametry.*

Řešení.

```
> f:=x->piecewise(x<0,a,b):
> GetFourierSeries(f(x),x=-Pi..Pi);
```

$$\frac{\pi b + \pi a}{2\pi} + \sum_{n=1}^{\infty} \frac{\left(\frac{a((-1)^n - 1)}{n} - \frac{b((-1)^n - 1)}{n}\right) \sin(nx)}{\pi}$$

Tento výsledek můžeme zjednodušit například procedurou `SimplifyCoefficients`.

```
> SimplifyCoefficients(%, simplify);
```

$$\frac{1}{2}a + \frac{1}{2}b + \sum_{n=1}^{\infty} \frac{((-1)^n - 1)(a-b)\sin(nx)}{\pi n}$$

□

Příklad 4.9. *Vypočítejte Fourierovu řadu funkce $y = x\cos x$ na intervalu $[-\pi, \pi]$. Pomocí animace znázorněte konvergenci řady k periodickému rozšíření funkce f.*

Řešení.

```
> f:=x->x*cos(x):
> FS:=GetFourierSeries(f(x),x=-Pi..Pi);
```

$$FS := -\frac{1}{2}\sin x + \sum_{n=2}^{\infty} \frac{2n(-1)^n \sin(nx)}{n^2 - 1}$$

Všimněme si, že pro $n = 1$ bychom dostali ve jmenovateli zlomku nulu. S tím jsme se již setkali u příkladů 4.3 a 4.4. Zde však procedura `GetFourierSeries` tento případ vyřešila sama, náš zásah tedy není nutný.

Nyní přistoupíme k vytvoření animace.

```
> gb:=DrawPeriodicExtension(f(x),x=-Pi..Pi,-6..6,drawlimits,
  color=aquamarine,thickness=2):
> anim:=seq(plots[display](gb, DrawPartialSum(FRada,i,-6..6,
  numpoints=250)), i=0..20):
> plots[display](anim, insequence=true,scaling=constrained);
```

Vybrané náhledy z animace jsou na obrázku 4.12. □

Příklad 4.10. *Funkci $y = \arcsin(\sin x)$ rozviňte v Fourierovu řadu.*

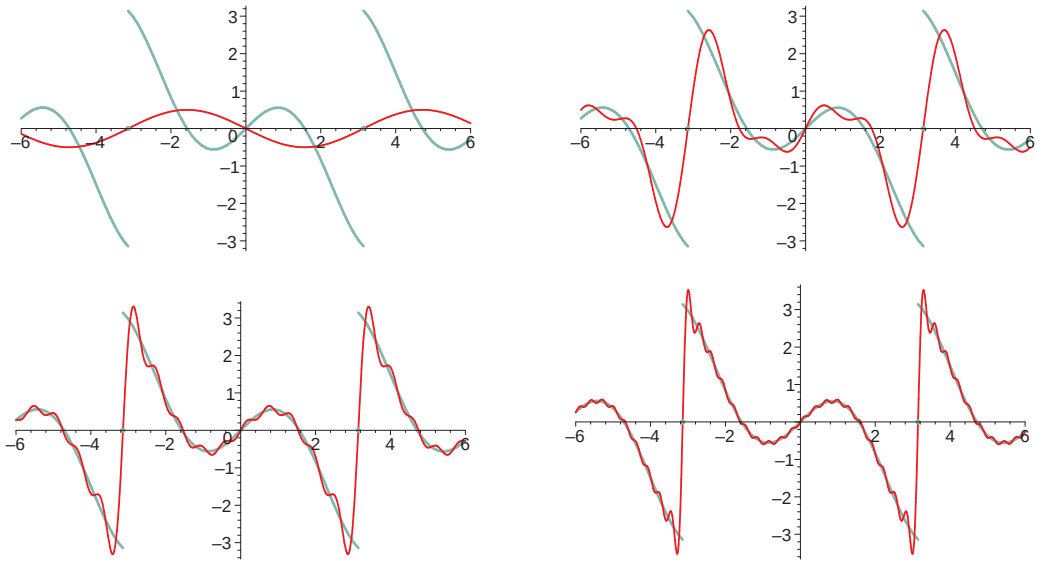
Řešení. Tentokrát zjistíme, že postup užívaný dříve nevede k cíli. Maple nedokáže spočítat požadované integrály. Zde je třeba Maplu asistovat a danou funkci nejdříve upravit. Stačí si uvědomit, že na intervalu $[-\pi, \pi]$ můžeme funkci f nahradit funkcí g s předpisem

$$g(x) = \begin{cases} -x - \pi & \text{pro } x \in [-\pi, -\frac{\pi}{2}) \\ x & \text{pro } x \in [-\frac{\pi}{2}, \frac{\pi}{2}) \\ \pi - x & \text{pro } x \in [\frac{\pi}{2}, \pi]. \end{cases}$$

V oboru reálných čísel je pak funkce f periodickým rozšířením funkce g. Lehce se o tom můžeme přesvědčit například z grafů obou funkcí.

Nyní již Fourierovu řadu spočítáme známým postupem.

```
> g:=x->piecewise(x<-Pi/2,-x-Pi,x<Pi/2,x,Pi-x):
```



Obr. 4.12: Vybrané náhledy z animace (částečné součty s_1, s_4, s_{10} a s_{20})

```
> GetFourierSeries(g(x),x=-Pi..Pi);
```

$$\sum_{n=1}^{\infty} \frac{\left(\frac{2\sin\left(\frac{1}{2}n\pi\right)+\cos\left(\frac{1}{2}n\pi\right)n\pi}{n^2} + \frac{2\sin\left(\frac{1}{2}n\pi\right)-\cos\left(\frac{1}{2}n\pi\right)n\pi}{n^2} \right) \sin(nx)}{\pi}$$

```
> SimplifyCoefficients(%, simplify);
```

$$\sum_{n=1}^{\infty} \frac{4\sin(\frac{1}{2}\pi n)\sin(nx)}{\pi n^2}$$

Poznamenejme ještě, že budeme-li se zabývat funkcí f pouze na intervalu $[-\frac{\pi}{2}, \frac{\pi}{2}]$, můžeme ji nahradit funkcí $g(x) = x$. Její Fourierova řada bude mít tvar

$$\sum_{n=1}^{\infty} -\frac{(-1)^n \sin(2nx)}{n}.$$

□

Příklad 4.11. *Spočítejte Fourierovu řadu funkce* $y = \operatorname{sgn} x$ *pro* $x \in [-\pi, \pi]$ *a určete funkční hodnoty částečných součtů* $s_n\left(\pi - \frac{\pi}{n}\right)$ *pro* $n = 10, 50, 100, 500, 1000, 5000, 10000$. *Dále nakreslete graf částečného součtu* s_{20} *a graf příslušného součtu získaného metodou* σ*-aproximace.*

Řešení. Při výpočtu funkčních hodnot částečných součtů Fourierovy řady využijeme proceduru `Evaluate` z knihovny `FourierTrigSeries`.

```
> f:=x->signum(x):
> FS:=GetFourierSeries(f(x), x=-Pi..Pi);
```

$$FS := \sum_{n=1}^{\infty} -\frac{2((-1)^n - 1)\sin(nx)}{\pi n}$$

```
> for i in [10,50,100,500,1000, 5000, 10000] do
> evalf(Evaluate(FS, x=(Pi-Pi/i), trunc=i));
> end do;
```

```
1.182328208
1.179113102
1.179013079
1.178981076
1.178980077
1.178979754
1.178979737
```

Všimněme si, že hodnota „překmitu“ se s rostoucím n významně nemění a činí téměř $18procent$.

Následně nadefinujeme funkci $\operatorname{sinc} x$, pomocí cyklu sestrojíme aproximovaný součet a vykreslíme oba požadované grafy.

```
> sinc:=x->piecewise(x=0,1,sin(x)/x):
> N:=20:
> s:=Coefficients(FS, 0):
> for i from 1 to (N-1) do
> s:=s+sinc(i*Pi/N)*Coefficients(FS,i)[2]*sin(i*x):
> end do:
> plot(GetPartialSum(FS,20), x=-2*Pi..2*Pi, numpoints=200,
  scaling=constrained);
> plot(s,x=-2*Pi..2*Pi,scaling=constrained,numpoints=200);
```

Grafy obou součtů jsou na obrázku 4.13 □

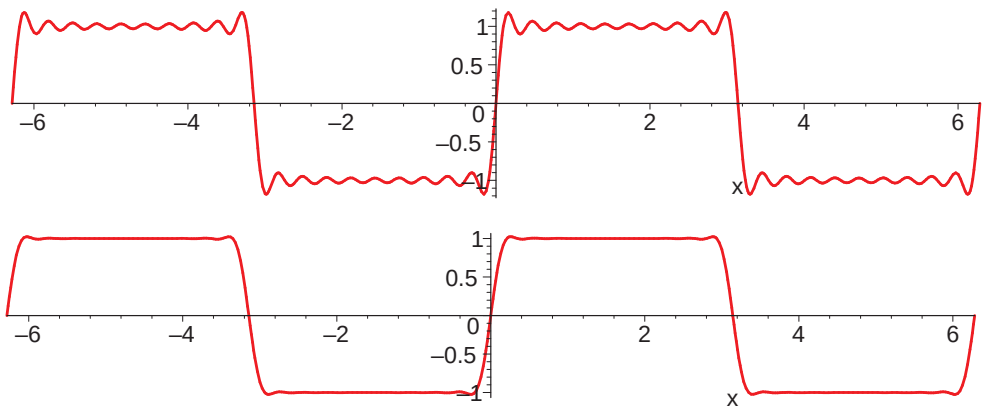
Příklad 4.12. *Spočítejte Fourierovu řadu funkce $y = x$ pro $x \in [-\pi, \pi]$ a s využitím Besselovy identity spočítejte kvadratickou odchylku částečných součtů stupňů 100, 500 a 1000.*

Řešení.

```
> FS:=GetFourierSeries(f(x), x=-Pi..Pi);
```

$$FS := \sum_{n=1}^{\infty} \frac{2(-1)^{(1+n)}\sin(nx)}{n}$$

Výpočet kvadratické odchylky pomocí dříve uvedené funkce `qdev`, která využívá vztahu (4.5), by byl časově i paměťově náročný. Níže uvedená procedura `qdevBE` využívá při výpočtu Besselovy identity.



Obr. 4.13: Částečný součet s_{20} a jeho σ-aproximace.

```
> qdevBE:=proc(f, var, N) local FS, L, normf, tmpsum, res, i;
> FS:=GetFourierSeries(f,var);
> normf:=int(f^2,var);
> L:=op(2, op(2,var)) - op(1, op(2,var));
> tmpsum:=L*Coefficients(FS,0)^2;
> for i from 1 to N do
> tmpsum:=tmpsum + L/2*(Coefficients(FS,i)[1]^2+Coefficients(FS,i)[2]^2);
> end do:
> res:=evalf(sqrt(normf - tmpsum));
> return res;
> end proc:
```

Nyní již můžeme přistoupit k požadovaným výpočtům.

```
> qdevBE(f(x),x=-Pi..Pi,100);
```

0.3536063890

```
> qdevBE(f(x),x=-Pi..Pi,500);
```

0.1584538800

```
> qdevBE(f(x),x=-Pi..Pi,1000);
```

0.1120718057

□

Nyní se vrátíme k úloze o kmitání struny z části 4.3.

Příklad 4.13. *Ilustrujte pomocí animace kmitání struny, jejíž krajní body jsou* $[0,0]$ *a* $[\pi,0]$ *a jejíž výchozí tvar při napnutí popisuje funkce*

$$\varphi(x) = \begin{cases} \frac{x}{2} & \text{pro } x \in [0, \frac{\pi}{2}) \\ \frac{\pi - x}{2} & \text{pro } x \in [\frac{\pi}{2}, \pi] \end{cases} .$$

Řešení. Vyjdeme ze závěrů učiněných v části 4.3. Nejdříve nadefinujeme funkci $\varphi(x)$ a její liché rozšíření.

```
> restart:
> with(FourierTrigSeries):
> phi:=x->piecewise(x<Pi/2 ,x/2,Pi/2-x/2):
```

Nakreslíme graf zachycující počáteční tvar struny. Zároveň graf uložíme do proměnné, později jej využijeme také při vytváření animace.

```
> g1:=plot(phi,0..Pi, scaling=constrained, thickness=2,
   color=aquamarine):%;
```

Spočítáme Fourierovu řadu lichého rozšíření funkce $\varphi(x)$.

```
> FS:=GetFourierSeries(phi(x), x=0..Pi, odd);
```

$$FS := \sum_{n=1}^{\infty} \frac{\left(\frac{\cos\left(\frac{1}{2}\pi n\right) n\pi + 2\sin\left(\frac{1}{2}\pi n\right)}{2n^2} - \frac{-2\sin\left(\frac{1}{2}\pi n\right) + \cos\left(\frac{1}{2}\pi n\right) n\pi}{2n^2} \right) \sin(nx)}{\pi}$$

```
> FS:=SimplifyCoefficients(FS, simplify);
```

$$FS := \sum_{n=1}^{\infty} \frac{2\sin\left(\frac{1}{2}\pi n\right)\sin(nx)}{\pi n^2}$$

Nyní z této řady vytvoříme pomocí procedury `Create` novou řadu tak, že koeficienty řady rozšíříme výrazem $\cos nt$.

```
> FS2:=Create(Coefficients(FS)*cos(n*t), SinTrigP(n,x));
```

$$FS := \sum_{n=1}^{\infty} \frac{2\sin\left(\frac{1}{2}\pi n\right)\cos(nt)\sin(nx)}{\pi n^2}$$

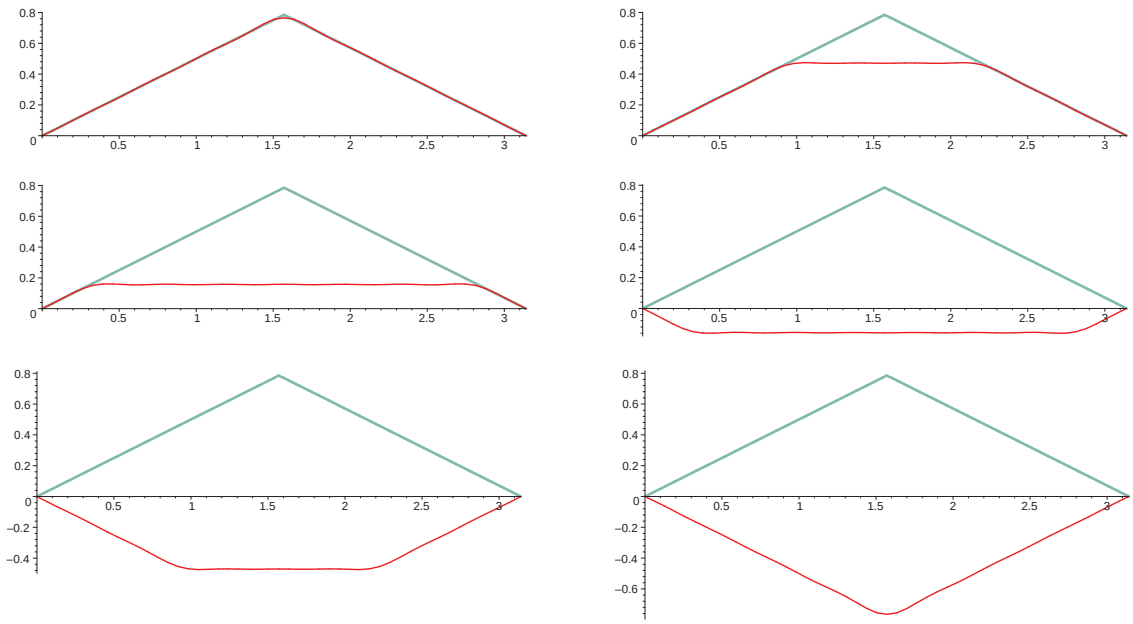
Pro aproximaci funkce $\varphi(x)$ použijeme prvních 15 členů řady.

```
> s:=GetPartialSum(FS2, 15);
```

$$\begin{aligned} s := & \frac{2\sin(x)\cos(t)}{\pi} - \frac{2}{9}\frac{\sin(3x)\cos(3t)}{\pi} + \frac{2}{25}\frac{\sin(5x)\cos(5t)}{\pi} - \frac{2}{49}\frac{\sin(7x)\cos(7t)}{\pi} \\ & + \frac{2}{81}\frac{\sin(9x)\cos(9t)}{\pi} - \frac{2}{121}\frac{\sin(11x)\cos(11t)}{\pi} + \frac{2}{169}\frac{\sin(13x)\cos(13t)}{\pi} \\ & - \frac{2}{225}\frac{\sin(15x)\cos(15t)}{\pi} \end{aligned}$$

Nyní již můžeme přikročit k sestrojení animace.

```
> with(plots):
> g2:=animate(s,x=0..Pi,t=0..2*Pi,frames=50, scaling=constrained):
> display(g1,g2);
```



Obr. 4.14: Vybrané náhledy z animace kmitající struny

Vybrané náhledy z animace jsou obrázku 4.14. Červenou barvou je znázorněna kmitající struna, zelenou pak graf funkce $\varphi(x)$.

Animaci s časovým průběhem prvních tří nenulových harmonických složek zobrazíme například následující sérií příkazů.

```
> H1:=GetPartialSum(FS2,1):
> H3:=GetPartialSum(FS2,3)-GetPartialSum(FS2,2):
> H5:=GetPartialSum(FS2,5)-GetPartialSum(FS2,4):
> animate(H1,H3,H5,x=0..Pi,t=0..2*Pi, frames=50);
```

□

Doporučujeme čtenáři vyzkoušet uvedený postup také s funkcemi

$$\phi(x) = \frac{1}{4}x(x-\pi) \quad \text{a} \quad \psi(x) = \begin{cases} x & \text{pro } x \in [0, \frac{\pi}{5}) \\ \frac{2\pi}{5} - x & \text{pro } x \in [\frac{\pi}{5}, \frac{2\pi}{5}) \\ 0 & \text{pro } x \in [\frac{2\pi}{5}, \pi] \end{cases}.$$

Příklad 4.14. *Metodou Fourierových řad řešte diferenciální rovnici*

$$y'' + 2y = \sum_{n=1}^{\infty} \frac{\sin nx}{n^4}.$$

Řešení. Budeme postupovat podobně jako v případě řad mocninných. Vytvoříme tedy obecnou reprezentaci Fourierovy řady a dosadíme ji do řešené diferenciální rovnice. Hodnoty koeficientů pak získáme porovnáním s řadou na pravé straně.

Nejdříve vytvoříme reprezentaci pravé strany rovnice. Využijeme již zmíněné procedury `Create`.

```
> RHS:=Create({[0], general=[0,1/n^4]}, CosSinTrigP(n,x));
```

$$RHS := \sum_{n=1}^{\infty} \frac{\sin nx}{n^4}$$

Následně vytvoříme obecnou reprezentaci Fourierovy řady.

```
> FS:=Create({[a0], 'general'=[aN,bN]}, CosSinTrigP(n,x));
```

$$FS := a0 + \sum_{n=1}^{\infty} (aN\cos(nx) + bN\sin(nx))$$

Nyní tuto řadu dvakrát derivujeme a k výsledku přičteme dvojnásobek řady samotné. Získáme tak levou stranu diferenciální rovnice po dosazení obecné Fourierovy řady.

```
> tmp:=Derivate(Derivate(FS, 'x'), 'x');
```

$$tmp := \sum_{n=1}^{\infty} \left(-n^2 aN\cos(nx) - n^2 bN\sin(nx)\right)$$

```
> LHS:=Add(tmp,FS, 1, 2);
```

$$LHS := 2a0 + \sum_{n=1}^{\infty} \left((2aN - n^2 aN)\cos(nx) + (2bN - n^2 bN)\sin(nx)\right)$$

Konečně odečteme pravou stranu rovnice od levé.

```
> S:=Add(LHS,RHS, 1, -1);
```

$$S := 2a0 + \sum_{n=1}^{\infty} \left((2aN - n^2 aN)\cos(nx) + \left(-\frac{1}{n^4} + 2bN - n^2 bN\right)\sin(nx)\right)$$

Získali jsme nekonečnou řadu, jejíž všechny koeficienty musí být rovny nule. V našem případě stačí pouze sestrojit a vyřešit soustavu rovnic pro neznámé aN a bN.

```
> Coefficients(S);
```

$$\left[2aN - n^2 aN, -\frac{1}{n^4} + 2bN - n^2 bN\right]$$

```
> {%[1]=0, %[2]=0};
```

$$\{2aN - n^2 aN = 0, -\frac{1}{n^4} + 2bN - n^2 bN = 0\}$$

```
> solve(%, {aN, bN});
```

$$\{aN = 0, bN = -\frac{1}{n^4(-2 + n^2)}\}$$

Konečně můžeme sestrojit hledané řešení.

```
> Y:=Create({subs(%, bN)}, SinTrigP(n,x));
```

$$Y := \sum_{n=1}^{\infty} \left(-\frac{\sin(nx)}{n^4(-2+n^2)} \right)$$

Nyní bychom měli ověřit, zda tato řada opravdu konverguje. Tento úkol přenecháme čtenáři, aplikací Weierstrassova kritéria lze dokázat stejnoměrnou konvergenci uvedené řady. Správnost řešení můžeme ověřit jeho dosazením do levé strany rovnice.

```
> Add(Derivate(Derivate(Y, 'x'), 'x'), Y, 1, 2);
```

$$\sum_{n=1}^{\infty} \left(-\frac{2}{n^4(-2+n^2)} + \frac{1}{n^2(-2+n^2)} \right) \sin(nx)$$

```
> SimplifyCoefficients(%, simplify);
```

$$\sum_{n=1}^{\infty} \frac{\sin nx}{n^4}$$

□

4.6 Fourierovy řady a jiné ortogonální systémy

Mimo ortogonální systém (4.1), jež byl zmíněn na začátku kapitoly, existují i další ortogonální systémy. Například systémy funkcí

$$1, \cos x, \cos 2x, \cos 3x, \ldots$$

a

$$\sin x, \sin 2x, \sin 3x, \ldots$$

jsou ortogonální na intervalu $[0, \pi]$. Prakticky jsme se s nimi již setkali při hledání Fourierovy řady sudého respektive lichého periodického rozšíření funkce definované na itervalu $[0, \pi]$.

Další často používané ortogonální systémy tvoří tzv. *ortogonální polynomy*. Tyto polynomy získáme ortogonalizací, respektive ortonormalizací systému polynomů

$$1, x, x^2, x^3, x^4, \ldots \tag{4.10}$$

Některé typy ortogonálních polynomů si nyní představíme. Pro naše úvahy však potřebujeme obecnější definici skalárního součinu dvou funkcí.

Definice. Buď w, f a g integrovatelné funkce na intervalu $I = [a, b]$, w je spojitá na I a $w(x) > 0$ pro $x \in I$. Číslo

$$(f, g) = \int_a^b w(x) f(x) g(x) \, \mathrm{d}x \tag{4.11}$$

nazveme skalárním součinem funkcí f a g.

Funkci $w(x)$ z předchozí definice nazýváme *váhovou funkcí*. Je patrné, že v případě $w(x) = 1$ pro $x \in \mathbb{R}$ a integrací přes interval $[-\pi, \pi]$ získáme skalární součin v podobě, v jaké jsme jej zavedli v předcházející kapitole.

Pro výpočty Fourierových řad funkcí vzhledem k těmto ortogonálním systémům použijeme následující tvrzení.

Věta. Buď $\{\varphi_n\}$ ortogonální posloupnost funkcí na intervalu $[a, b]$, $\{c_n\}$ posloupnost reálných čísel. Nechť řada

$$\sum_{n=1}^{\infty} c_n \varphi_n(x)$$

stejnoměrně konverguje k funkci f na intervalu $[a, b]$. Pak pro konstanty c_n $(n \in \mathbb{N})$ platí:

$$c_n = \frac{(f, \varphi_n)}{(\varphi_n, \varphi_n)} = \frac{(f, \varphi_n)}{\|\varphi_n\|^2} \tag{4.12}$$

Definice. Buď $\{\varphi_n\}$ ortogonální posloupnost funkcí na intervalu $[a, b]$, f integrovatelná funkce na $[a, b]$. Pak čísla c_n vyjádřená vzorcem (4.12) nazýváme *Fourierovy koeficienty funkce f* vzhledem k ortogonální posloupnosti $\{\varphi_n\}$ a řadu

$$\sum_{n=1}^{\infty} c_n \varphi_n,$$

kde c_n jsou Fourierovy koeficienty, *Fourierovou řadou funkce f* vzhledem k ortogonální posloupnosti $\{\varphi_n\}$.

Poznámka. V případě, kdy posloupnost $\{\varphi_n\}$ je ortonormální, platí pro Fourierovy koeficienty funkce f jednodušší vztah $c_n = (f, \varphi_n)$.

Také v tomto případě je přiřazení Fourierovy řady k dané funkci pouze formální, neboť nevíme, zda tato řada vůbec konverguje.

Odvození jednotlivých typů ortogonálních polynomů lze provést několika způsoby. My využijeme již zmíněný výpočet ortonormalizací systému (4.10) použijeme následující proceduru `GramSchmidt`. Procedura provádí ortonormalizaci konečného systému lineárně nezávislých funkcí Gram-Schmidtovým ortonormalizačním procesem. Proceduře se předávají dva parametry. Prvním je již zmíněný seznam lineárně nezávislých funkcí, druhým je funkce představující konkrétní podobu skalárního součinu.

```
> GramSchmidt:=proc(base, inner_product) local ortonormbase, norm, i, proj,
  new_vector;
> ortonormbase:=[];
> norm:=a->sqrt(inner_product(a,a));
> for i from 1 to nops(base) do
> if i = 1 then
> ortonormbase:=[op(ortonormbase), base[i]/norm(base[i])];
```

```
> else
> proj:=sum(inner_product(base[i],
  ortonormbase[j])/norm(ortonormbase[j])*ortonormbase[j], j=1..(i-1));
> new_vector:=base[i]-proj;
> ortonormbase:=[op(ortonormbase), simplify(new_vector/norm(new_vector))];
> end if;
> end do;
> RETURN(ortonormbase);
> end:
```

Legendrovy polynomy

Ortonormalizací systému (4.10) na intervalu $[-1, 1]$ s váhovou funkcí $w(x) = 1$ pro $x \in \mathbb{R}$ získáme, až na násobek konstantou, tzv. *Legendrovy polynomy.*

Příklad 4.15. *S využitím procedury* `GramSchmidt` *nalezněte prvních pět Legendrových polynomů a vykreslete jejich grafy. Využijte přitom skutečnosti, že funkční hodnota v bodě* $x = 1$ *je v případě Legendrových polynomů rovna jedné.*

Řešení. Definujeme funkci představující skalární součin.

```
> f:=(a,b)->int(a*b,x=-1..1);
```

Ortonormalizujeme polynomy 1, x, x^2, x^3 a x^4.

```
> GramSchmidt([1, x, x^2, x^3, x^4], f);
```

$$\left[\frac{\sqrt{2}}{2}, \frac{x\sqrt{6}}{2}, \frac{(3x^2-1)\sqrt{10}}{4}, \frac{x(5x^2-3)\sqrt{14}}{4}, \frac{3(35x^4-30x^2+3)\sqrt{2}}{16}\right]$$

Získali jsme normované Legendrovy polynomy. Nyní jednotlivé normované polynomy vydělíme konstantou představující funkční hodnotu polynomu v bodě $x = 1$, abychom splnili podmínku $P_n(1) = 1$.

```
> L:=map(pol->expand(pol/eval(pol,x=1)), %);
```

$$L := \left[1, x, \frac{3x^2}{2} - \frac{1}{2}, \frac{5}{2}x^3 - \frac{3}{2}x, \frac{35}{8}x^4 - \frac{15}{4}x^2 + \frac{3}{8}\right]$$

Můžeme ověřit ortogonalitu vybraných polynomů, například:

```
> pol:=L[3];
```

$$pol := \frac{3x^2}{2} - \frac{1}{2}$$

```
> for pol2 in L do
> 'f'(pol, pol2) = f(pol,pol2);
```

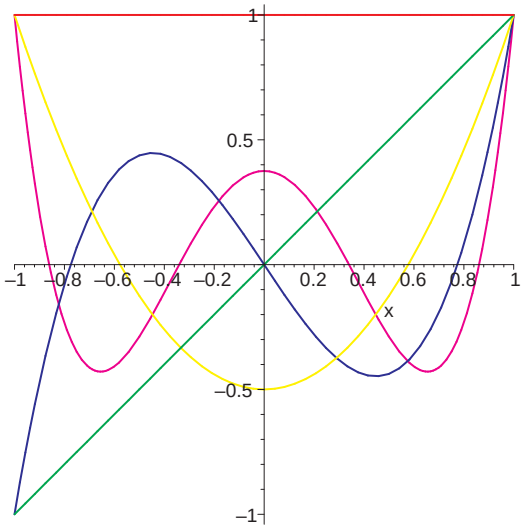
```
> end do;
```

$$f\left(\frac{3x^2}{2}-\frac{1}{2},1\right)=0$$
$$f\left(\frac{3x^2}{2}-\frac{1}{2},x\right)=0$$
$$f\left(\frac{3x^2}{2}-\frac{1}{2},\frac{3x^2}{2}-\frac{1}{2}\right)=\frac{2}{5}$$
$$f\left(\frac{3x^2}{2}-\frac{1}{2},\frac{5}{2}x^3-\frac{3}{2}x\right)=0$$
$$f\left(\frac{3x^2}{2}-\frac{1}{2},\frac{35}{8}x^4-\frac{15}{4}x^2+\frac{3}{8}\right)=0$$

Na závěr vykreslíme grafy nalezených polynomů. Výsledný graf je na obrázku 4.15.

```
> plot(L, x=-1..1, scaling=constrained);
```

□



Obr. 4.15: Prvních pět Legendrových polynomů na intervalu $[-1,1]$.

Výpočet Legendrových polynomů ortogonalizací systému (4.10) je značně nepraktický. Existuje však celá řada indentit, které lze pro výpočet Legendrových polynomů využít. Uveďme například tzv. *Rodriguezův vzorec*

$$P_n(x)=\frac{1}{2^n n!}\frac{\mathrm{d}^n}{\mathrm{d}x^n}\left(x^2-1\right)^n$$

a rekurentní formuli

$$P_{n+1}(x)=\frac{2n+1}{n+1}xP_n(x)-\frac{n}{n+1}P_{n-1}(x).$$

Navíc Maple samotný nabízí procedury pro jejich výpočet Legendrových polynomů.

Příklad 4.16. *Napiště procedury či funkce počítající Legendrovy polynomy pomocí dvou výše uvedených vztahů. Srovnejte získané výsledky z výstupy procedur nabízených programem Maple.*

Řešení. V případě Rodriguezova vzorce je řešení přímočaré.

```
> Leg1:=proc(n::nonnegint,x);
> if n=0 then 1
> else expand(1/(2^n*n!)*diff((x^2-1)^n,x$n));
> end if;
> end proc:
```

Rekurentní formuli nejdříve upravíme substitucí $k = n + 1$ na vhodnější tvar.

```
> P(n+1,x)=(2*n+1)/(n+1)*x*P(n,x)-n/(n+1)*P(n-1,x);
```

$$P(n+1,x) = \frac{(2n+1)xP(n,x)}{n+1} - \frac{nP(n-1,x)}{n+1}$$

```
> algsubs(n+1=k,%);
```

$$P(k,x) = \frac{xP(k-1,x)(2k-1)}{k} - \frac{P(k-2,x)(k-1)}{k}$$

Nyní již samotná definice procedury.

```
> Leg2:=proc(k::nonnegint,x) option remember;
> if k=0 then 1
> elif k=1 then x
> else expand((2*k-1)/k*x*Leg2(k-1,x)-(k-1)/k*Leg2(k-2,x));
> end if;
> end proc:
```

Pomocí procedur spočítame Legendrovy polynomy stupně 8 a 9.

```
> Leg1(8,x);
```

$$\frac{6435}{128}x^8 - \frac{3003}{32}x^6 + \frac{3465}{64}x^4 - \frac{315}{32}x^2 + \frac{35}{128}$$

```
> Leg2(9,x);
```

$$\frac{12155}{128}x^9 - \frac{6435}{32}x^7 + \frac{9009}{64}x^5 - \frac{1155}{32}x^3 + \frac{315}{128}x$$

Pro přímý výpočet Legendrových polynomů můžeme v programu Maple použít proceduru `LegendreP` nebo proceduru `P`, jež je součástí knihovny `orthopoly`.[17] Tyto dvě procedury použijeme pro kontrolu správnosti předchozích dvou výsledků.

```
> expand(LegendreP(9,x));
```

$$\frac{6435}{128}x^8 - \frac{3003}{32}x^6 + \frac{3465}{64}x^4 - \frac{315}{32}x^2 + \frac{35}{128}$$

[17] Knihovna `orthopoly` je v nápovědě označovaná jako zastaralá, v novějších vydáních programu Maple proto nemusí být dostupná. Místo knihovny `orthopoly` se doporučuje používat právě proceduru `LegendreP` a procedury podobné.

```
> with(orthopoly);
```

$$[G, H, L, P, T, U]$$

```
> P(8,x);
```

$$\frac{12155}{128}x^9 - \frac{6435}{32}x^7 + \frac{9009}{64}x^5 - \frac{1155}{32}x^3 + \frac{315}{128}x$$

□

V následujícím příkladu budeme ilustrovat výpočet částečného součtu Fourierovy řady vzhledem k ortogonálnímu systému Legendrových polynomů, tzv. *Fourierovy-Legendrovy řady*. Narozdíl od ortogonálního systému tvořeného funkcemi $\sin nx$ a $\cos nx$ nedovedeme získat předpis Fourierovy řady v uzavřeném tvaru a jsme tak odkázáni pouze na její částečné součty.

Příklad 4.17. *Spočítejte první tři nenulové koeficienty Fourierovy-Legendrovy řady funkce* $\sin \pi x$ *na intervalu [-1,1]. Dále nakreslete graf získaného částečného součtu spolu s grafem vyšetřované funkce.*

Řešení. Definujeme vyšetřovanou funkci a také funkci pro reprezentaci skalárního součinu.

```
> f:=x->sin(Pi*x):
> g:=(a,b)->int(a*b,x=-1..1):
```

Dále definujeme funkci pro výpočet koeficientů Fourierovy řady podle vzorce (4.12).

```
> cn:=n->g(LegendreP(n,x),f(x))/g(LegendreP(n,x),LegendreP(n,x));
```

$$cn := n \to \frac{g(LegendreP(n,x), f(x))}{g(LegendreP(n,x), LegendreP(n,x))}$$

Protože funkce $\sin \pi x$ je funkce lichá, bude Fourierova řada tvořena pouze lichými mocninami. Spočítejme tedy prvních 6 koeficientů Fourierovy řady.

```
> N:=6:
> for i from 0 to (N-1) do
> print(c[i]=cn(i));
> end do;
```

$$c_0 = 0$$
$$c_1 = \frac{3}{\pi}$$
$$c_2 = 0$$
$$c_3 = \frac{7(-15+\pi^2)}{\pi^3}$$
$$c_4 = 0$$
$$c_5 = \frac{11(\pi^4-105\pi^2+945)}{\pi^5}$$

Částečný součet řady má tedy následující tvar.

```
> S:=sum(cn(k)*LegendreP(k,x),k=0..N-1);
```

$$S := \frac{3x}{\pi} + \frac{7(-15+\pi^2)LegendreP(3,x)}{\pi^3} + \frac{11(\pi^4-105\pi^2+945)LegendreP(5,x)}{\pi^5}$$

```
> pol:=expand(S);
```

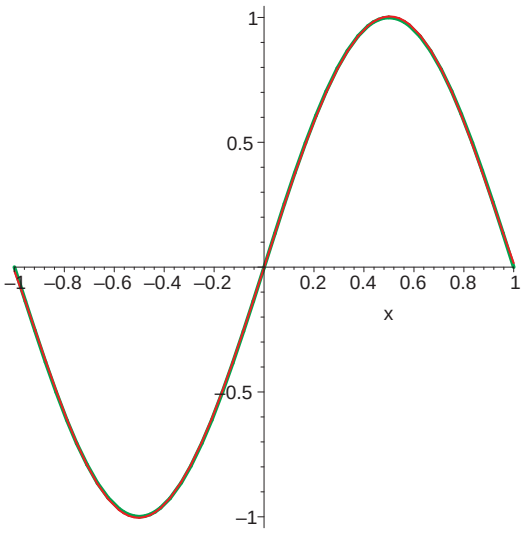
$$\frac{105x}{8\pi}+\frac{39375x^3}{4\pi^3}-\frac{16065x}{8\pi^3}-\frac{315x^3}{4\pi}+\frac{693x^5}{8\pi}-\frac{72765x^5}{8\pi^3}+\frac{654885x^5}{8\pi^5}-\frac{363825x^3}{4\pi^5}+\frac{155925x}{8\pi^5}$$

```
> evalf(pol);
```

$$3.10346042x - 4.81438833x^3 + 1.7269051x^5$$

Konečně sestrojíme grafy obou funkcí.

```
> g1:=plot(f(x),x=-1..1,thickness=4,color=green,scaling=constrained):
> g2:=plot(pol,x=-1..1,thickness=2,color=red,scaling=constrained):
> plots[display](g2,g1);
```



Obr. 4.16: Funkce $\sin \pi x$ a její aproximace Legendrovými polynomy.

Výsledný graf je na obrázku 4.16. Přesnost aproximace je až zarážející. Zajímavé je také srovnání s grafem Maclaurinova polynomu odpovídajícího stupně. Jeho graf je na obrázku 4.17.

```
> pol2:=convert(taylor(f(x),x=0,N), polynom);
```

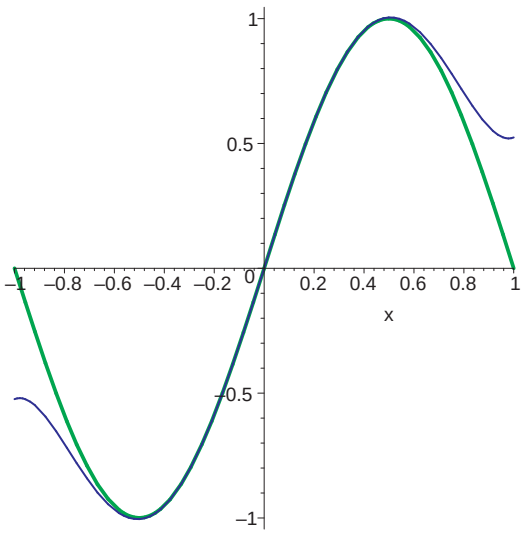
$$pol := \pi x - \frac{1}{6}\pi^3 x^3 + \frac{1}{120}\pi^5 x^5$$

```
> g3:=plot(tayl,x=-1..1,thickness=2,color=blue,scaling=constrained):
> plots[display](g3,g1);
```

□

Čebyševovy polynomy prvního a druhého druhu

V případě Čebyševových polynomů prvního a druhého druhu má váhová funkce tvar $w(x) = \frac{1}{\sqrt{1-x^2}}$ respektive $w(x) = \sqrt{1-x^2}$.



Obr. 4.17: Funkce $\sin \pi x$ a její aproximace Maclaurinovým polynomem pátého stupně.

Příklad 4.18. *Nalezněte prvních pět Čebyševových polynomů prvního a druhého druhu a nakreslete jejich grafy.*

Řešení. Pro přímý výpočet Čebyševových polynomů prvního a druhého druhu můžeme využít procedury `ChebyshevT` respektive `ChebyshevU`. Přesto úlohu řešme opět ortogonalizací systému (4.10) pomocí procedury `GramSchmidt`.

Budeme tedy postupovat jako v případě Legendrových polynomů. Pro normalizaci získaných normovaných polynomů využijeme v případě Čebyševových polynomů prvního druhu opět identitu $T_n(1) = 1$.

Nejdříve definujeme skalární součin.

```
> g:=(a,b)->int((1-x^2)^(-1/2)*a*b,x=-1..1);
```

$$g := (a, b) \to \int_{-1}^{1} \frac{ab}{\sqrt{1-x^2}}\,\mathrm{d}x$$

S využitím procedury `GramSchmidt` spočítáme prvních 5 polynomů které normalizujeme a vykreslíme jejich grafy.

```
> N:=5:
> GramSchmidt([seq(x^n, n=0..N-1)], g);
```

$$\left[\frac{1}{\sqrt{\pi}}, \frac{x\sqrt{2}}{\sqrt{\pi}}, \frac{(2x^2-1)\sqrt{2}}{\sqrt{\pi}}, \frac{x(4x^2-3)\sqrt{2}}{\sqrt{\pi}}, \frac{(8x^4-8x^2+1)\sqrt{2}}{\sqrt{\pi}}\right]$$

```
> L:=map(pol->expand(pol/eval(pol,x=1)), %);
```

$$L := \left[1, x, 2x^2-1, 4x^3-3x, 8x^4-8x^2+1\right]$$

```
> plot(L,x=-1..1,thickness=2);
```

V případě Čebyševových polynomů druhého druhu postupujeme obdobně, polynomy normalizujeme pomocí identity $U_n(1) = n + 1$.

```
> h:=(a,b)->int((1-x^2)^(1/2)*a*b,x=-1..1);
```

$$h := (a, b) \to \int_{-1}^{1} \sqrt{1 - x^2}\, ab \, \mathrm{d}x$$

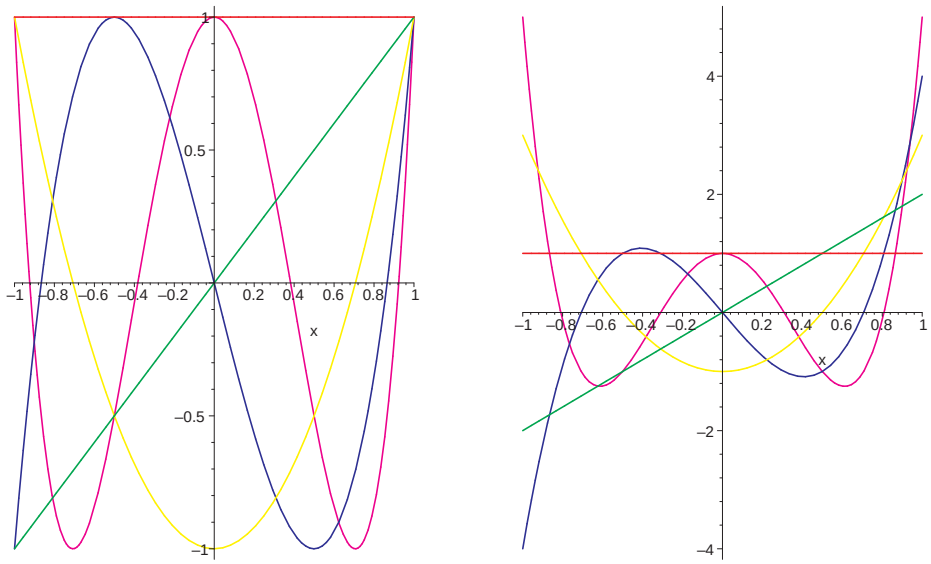
```
> N:=5:
> GramSchmidt([seq(x^n, n=0..N-1)], h);
```

$$\left[\frac{\sqrt{2}}{\sqrt{\pi}}, \frac{2x\sqrt{2}}{\sqrt{\pi}}, \frac{(4x^2-1)\sqrt{2}}{\sqrt{\pi}}, \frac{4x(2x^2-1)\sqrt{2}}{\sqrt{\pi}}, \frac{(16x^4-12x^2+1)\sqrt{2}}{\sqrt{\pi}}\right]$$

```
> L2:=expand([seq(%[i]/eval(%[i],x=1)*i,i=1..N)]);
```
[18]

$$L2 := [1, 2x, 4x^2 - 1, 8x^3 - 4x, 16x^4 + 1 - 12x^2]$$

```
> plot(L2,x=-1..1,thickness=2);
```



Obr. 4.18: Prvních pět Čebyševových polynomů prvního a druhého stupně.

Výsledné grafy Čebyševových polynomů jsou na obrázku 4.18. □

V případě Fourierových řad vzhledem k systému Čebyševových polynomů bývají tyto řady označovány jako *Fourierovy-Čebyševovy*.

Příklad 4.19. *Na intervalu* $[-1, 1]$ *aproximujte funkci* $y = \mathrm{e}^x$ *polynomem třetího stupně, jež je částečným součtem Fourierovy-Čebyševovy řady vzhledem k systému Čebyševových polynomů prvního druhu.*

[18] Skutečnost, že prvky v seznamu jsou indexovány od jedné, může ve výrazu působit matoucím dojmem.

Řešení. Budeme postupovat podobně jako v případě Legendrových polynomů.

```
> f:=x->exp(x):
> g:=(a,b)->int(a*b/sqrt(1-x^2),x=-1..1):
> n:=n->g(ChebyshevT(n,x),f(x))/g(ChebyshevT(n,x),ChebyshevT(n,x));
```

$$cn := n \to \frac{g(ChebyshevT(n,x), f(x))}{g(ChebyshevT(n,x), ChebyshevT(n,x)}$$

```
> N:=4:
> S:=sum(cn(k)*ChebyshevT(k,x),k=0..N-1):
```

V tomto případě Maple nedovede symbolicky vyhodnotit hodnoty integrálů, jež se objevují při výpočtu hodnot koeficientů Fourierovy-Čebyševovy řady.[19] Použijeme proto proceduru `evalf` pro získání numerické aproximace.

```
> pol:=evalf(expand(S));
```

$$pol := 0.9945705384 + 0.9973076585x + 0.5429906792x^2 + 0.1773473994x^3$$

Nakreslíme graf získaného polynomu a aproximované funkce. Výsledný graf je na obrázku 4.19.

```
> g1:=plot(f(x),x=-1..1,color=green,thickness=3):
> g2:=plot(pol,x=-1..1,color=red,thickness=2):
> plots[display](g2,g1);
```

Zůstaňme ještě chvíli u získaného výsledku. Aproximace Čebyševovými polynomy prvního druhu souvisí s problémem hledání tzv. *minimax polynomu*. Jedná se o polynom, jež danou funkci aproximuje nejlépe mezi všemi polynomy stejného stupně, přičemž hodnotícím kritériem je suprémová metrika, tj. $\sup\{|f(x) - P(x)|; x \in I\}$. Polynom minimax se špatně konstruuje. Používá se tzv. *Remezův algoritmus*, který postupně iteruje polohu bodů s extrémy $f(x) - P(x)$ (současně se tedy mění polohy uzlů, kde se $P(x) = f(x)$) tak, že interpolační polynom konverguje k polynomu minimax. Jako počáteční aproximace v Remezově algoritmu se často používá právě aproximace funkce f pomocí Čebyševových polynomů prvního druhu.

V programu Maple je Remezův algoritmus implementován procedurou `minimax` z knihovny `numapprox`. Spočítáme tedy aproximaci minimax polynomu třetího stupně.[20]

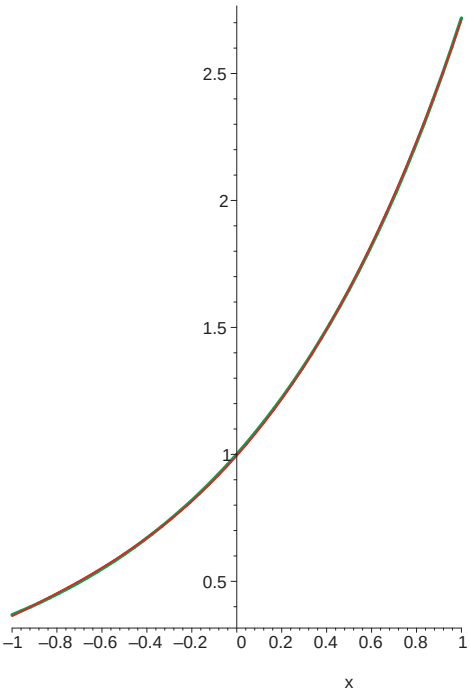
```
> minmax_f:=numapprox[minimax](f, -1..1, N-1):
> minmaxpol:=expand(minmax_f(x));
```

$$minmaxpol := 0.9945718793 + 0.9956674823x + 0.5429879654x^2 + 0.1795337116x^3$$

Pro srovnání spočítejme maximum chyby aproximace funkce e^x polynomem minimax a Čebyševovými polynomy.

[19] Z důvodu přílišné délky není výsledek předchozího příkazu uveden.

[20] Více informací o proceduře `minimax` a jejím použití získáme zadáním příkazu `?minimax`.



Obr. 4.19: Aproximace funkce e^x Čebyševovými polynomy.

```
> evalf(maximize(abs(minmaxpol-f(x)),x=-1..1));
```

$$0.0055359407$$

```
> evalf(maximize(abs(pol-f(x)),x=-1..1));
```

$$0.006065553$$

Z výsledku je patrné, že již samotná aproximace Čebyševovými polynomy je velmi dobrou aproximací a v případě funkce e^x je chyba aproximace pouze v řádu tisícin. □

Na závěr se krátce zmiňme o standardní knihovně `OrthogonalSeries`, která obsahuje procedury pro manipulaci s řadami klasických ortogonálních polynomů, či obecněji hypergeometrických polynomů. Používání knihovny je podobné již představené knihovně `FourierTrigSeries`, proto jej zde nebudeme podrobně rozebírat a čtenáře odkážeme na integrovanou nápovědu programu Maple.

Příklad 4.20. *Vyjádřete polynom $x^4-x^3-x^2-x+1$ jako lineární kombinaci Čebyševových polynomů druhého druhu.*

Řešení. Úlohu je možné řešit více způsoby. Můžeme se například vydat cestou řešení soustavy 5 rovnic o 5 neznámých a nebo můžeme spočítat Fourierovy řadu daného polynomů, která bude mít v tomto případě konečný rozvoj. S využitím knihovny `OrthogonalSeries`, konkrétně procedury `ChangeBasis` je řešení přímočaré.

```
> OrthogonalSeries[ChangeBasis](x^4-x^3-x^2-x+1, ChebyshevU(n,x));
```

$$-\frac{3}{4}ChebyshevU(1,x)+\frac{7}{8}ChebyshevU(0,x)+\frac{1}{16}ChebyshevU(4,x)-$$
$$-\frac{1}{16}ChebyshevU(2,x)-\frac{1}{8}ChebyshevU(3,x)$$

Zpětným roznásobením ověříme správnost výsledku.

```
> expand(OrthogonalSeries[ConvertToSum](%));
```

$$x^4-x^3-x^2-x+1$$

Uvedený postup nelze použít pro hledání Fourierových koeficientů jiných funkcí než polynomů. □

Knihovna `OrthogonalSeries` nepracuje přímo s Legendrovými polynomy, jelikož tyto polynomy nemá ve své databázi. Místo nich lze však použít *Jacobiho polynomy*, přičemž při výpočtu využijeme identitu `LegendreP(n,x) = JacobiP(n,0,0,x)`.

Závěr

Cílem práce bylo demonstrovat možnosti programu Maple při výuce nekonečných funkčních řad a posloupností a naprogramovat nové procedury, které automatizují, či alespoň výrazně usnadní, provádění souvisejících výpočtů. Její přínos lze spatřovat ve třech bodech.

Prvním je ilustrace řešení vybraných typů úloh pomocí programu Maple. V tomto ohledu navazuje zejména na již zmíněné publikace [6], [12], [18] a přináší několik nových témat.

Dále se jedná o automatizaci řešení úloh pomocí nových procedur. Tyto podstatně rozšiřují spektrum řešitelných úloh (jako příklad uveďme implementaci konvergenčních kritérií, jež nejsou omezeny pouze na mocninné řady). Podstatnou část práce tvoří programová knihovna `FourierTrigSeries`, jež slouží k počítání Fourierových trigonometrických řad reálných funkcí. Oproti jiným knihovnám věnovaným Fourierovým řadám umožňuje manipulovat během výpočtu s Fourierovou řadou jako celkem. Tento přístup byl inspirován standardní knihovnou `OrthogonalSeries` pro operace s řadami ortogonálních polynomů.

Posledním bodem je internetová prezentace knihovny `FourierTrigSeries`, která umožňuje provádět výpočty Fourierových řad a jejích koeficientů online prostřednictvím internetového prohlížeče. V současné době na stránkách každý měsíc registruji opakované přístupy z více než dvaceti zemí světa.

Na výsledky práce lze navázat zejména rozšířením množství a spektra řešených či neřešených příkladů. A to jak příkladů řešených standardními prostředky programu Maple, tak s pomocí nově naprogramovaných procedur či knihovny `FourierTrigSeries`. Takto řešené úlohy také přináší zpětnou vazbu a poukazují na mezery a chyby v návrhu procedur. Jejich znalost je nezbytným předpokladem pro další vývoj.

Práce pro mě byla přínosem zejména v získání hlubších zkušeností v práci s programem Maple. Knihovna `FourierTrigSeries` je bezpochyby nejrozsáhlejší kus programového kódu, který jsem v programu Maple napsal. Knihovnu se pokouším stále udržovat a rozvíjet.

Literatura

[1] Aratyn H. *A Short Course in Mathematical Methods With Maple*, World Scientific Publishing Company, Singapore, 2005, p. 716, ISBN 9812565957, 9789812565952

[2] Askey R. *Orthogonal Polynomials and Special Functions*, Society for Industrial and Applied Mathematics, Philadelphia, 1975, p. 118, ISBN 0898710189, 9780898710182

[3] Davis H. F. *Fourier Series and Orthogonal Functions*, Dover Publications, Inc., New York, 1989, p. 403, ISBN 0486659739

[4] Došlá Z., Novák V. *Nekonečné řady*, Masarykova Univerzita v Brně, Brno 2002

[5] Došlá Z., Plch R., Sojka P. *CD-ROM Diferenciální počet funkcí více proměnných,* edice *Matematická analýza s programem Maple*, `http://www.math.muni.cz/~plch/mapm/`, Masarykova Univerzita v Brně, Brno, 1999, ISBN 80-210-2203-5

[6] Došlá Z., Plch R., Sojka P. *CD-ROM Nekonečné řady,* edice *Matematická analýza s programem Maple*, `http://www.math.muni.cz/~plch/nkpm/`, Masarykova Univerzita v Brně, Brno, 2002, ISBN 80-210-3005-4

[7] Grebenča M. K. *Učebnice matematické analysy 2*, Nakladatelství Československé akademie věd, Praha, 1955

[8] Hardy G. H. , Rogosinski W. W. *Fourierovy řady*, SNTL, Praha 1971

[9] Jarník V. *Diferenciální počet II*, Academia, Praha, 1974

[10] Koukal S., Křížek M., Potůček R. *Fourierovy trigonometrické řady a metoda konečných prvků v komplexním oboru*, Academia, Praha, 2002

[11] Kreyszig E. *Advanced Engineering Mathematics 9th Edition*, Wiley, 2005, p. 1248, ISBN 0471488852, 9780471488859

[12] Kříž P. *Mocninné řady s programem Maple*, rigorózní práce, `http://www.math.muni.cz/~kriz/pseries/`, Masarykova univerzita v Brně, Brno, 2006, 46 s

[13] Kříž P., Šrot K. *Integrální počet v* $\mathbb{R}$ *s programem Maple*, Učitel matematiky, Jednota českých matematiků a fyziků, svazek 15, číslo 1, Praha, 2006, od s. 19-25, 7 s, ISSN 1210-9037

[14] Novák V. *Nekonečné řady*, skriptum UJEP, Brno, 1981

[15] Richards D. *Advanced Mathematical Methods with Maple*, Cambridge University Press, Cambridge, 2001, p. 878, ISBN 0521779812, 978-0521779814

[16] Russev P. *Analytic Functions and Classical Orthogonal Polynomials*, Publishing House of the Bulgarian Academy of Science, Sofia, 1984, p. 131

[17] Škrášek J. *Matematika II B – Fourierovy řady, obyčejné a parciální diferenciální rovnice*, SNTL, Praha, 1979, 184 s

[18] Šrot K. *Fourierovy řady s programem Maple*, rigorózní práce, `http://www.math.muni.cz/~xsrot/frady/`, Masarykova univerzita v Brně, Brno, 2005, 57 s

[19] Šrot K. *Knihovna FourierSeries pro manipulaci s trigonometrickými řadami v programu Maple*, Sborník příspěvků 3. konference Užití počítačů ve výuce matematiky, Jihočeská univerzita v Českých Budějovicích, České Budějovice, 2007, od s. 239-244, 6 s. ISBN 9788073940485

[20] Šrot K. *Technologie MapleNet a Fourierovy řady*, Pedagogický software 2006, Scientific Pedagogical Publishing, České Budějovice, 2006, od s. 222-224, 3 s, ISBN 8085645564

Elektronické zdroje

[21] Stone P. *Peter Stone's Maple Worksheets*, `http://www.peterstone.name/Maplepgs/maple_index.html`

[22] Hušek M., Pyrih P. *Matematická analýza 1, 2, 3, 4*, `http://matematika.cuni.cz/dl/analyza/`

[23] *Matematika online III*, `http://mathonline.fme.vutbr.cz/Matematika-III/sc-7-sr-1-a-26/default.aspx`

[24] Israel R. B. *Maple Advisor Database*, `http://www.math.ubc.ca/~israel/advisor/`

Elektronické zdroje z kolekce MapleApps

[25] Mathews J., Howel R. *Uniform Convergence*, `http://www.maplesoft.com/Applications/app_center_view.aspx?AID=1443&CID=14&SCID=121`

[26] DeVore C. *Animation of Taylor and Maclaurin series converging to their generated functions*, `http://www.maplesoft.com/Applications/app_center_view.aspx?AID=49`

[27] Anonymní *University of Wisconsin-Milwaukee, Dept. of Mathematics - Taylor series*, http://www.maplesoft.com/Applications/app_center_view.aspx?AID=685

[28] May M. *Taylor polynomials in several variables*, http://www.maplesoft.com/Applications/app_center_view.aspx?AID=705

[29] Herod J. *Taylor, Legendre, and Bernstein polynomials*, http://www.maplesoft.com/Applications/app_center_view.aspx?AID=50 Aproximace funkce pomocí Taylorových, Legendrových a Bernsteinových polynomů

[30] Lopez R. J. *Classroom Tips and Techniques: Teaching Fourier Series with Maple*, http://www.maplesoft.com/Applications/app_center_view.aspx?AID=2026, http://www.maplesoft.com/Applications/app_center_view.aspx?AID=2054, http://www.maplesoft.com/Applications/app_center_view.aspx?AID=2063, http://www.maplesoft.com/Applications/app_center_view.aspx?AID=2076

[31] Werner W. *Symbolic Computation of Fourier Series*, http://www.maplesoft.com/Applications/app_center_view.aspx?AID=185

[32] Khanshan A. H. *The Fourier Series package for Maple*, http://www.maplesoft.com/Applications/app_center_view.aspx?AID=2035

[33] Dzhamay A. *Fourier Series*, http://www.maplesoft.com/Applications/app_center_view.aspx?AID=1325

[34] Nakamura Y. *Fourier Series Expansion*, http://www.maplesoft.com/Applications/app_center_view.aspx?AID=2251

[35] Dzhamay A. *The Heat Equation: Separation of variables and Fourier series*, http://www.maplesoft.com/Applications/app_center_view.aspx?AID=1320

[36] Canright D. *Maple in Mathematics Education II: Fourier Series & Wave Equation, 1-D Wave Equation Solutions*, http://www.maplesoft.com/Applications/app_center_view.aspx?AID=187