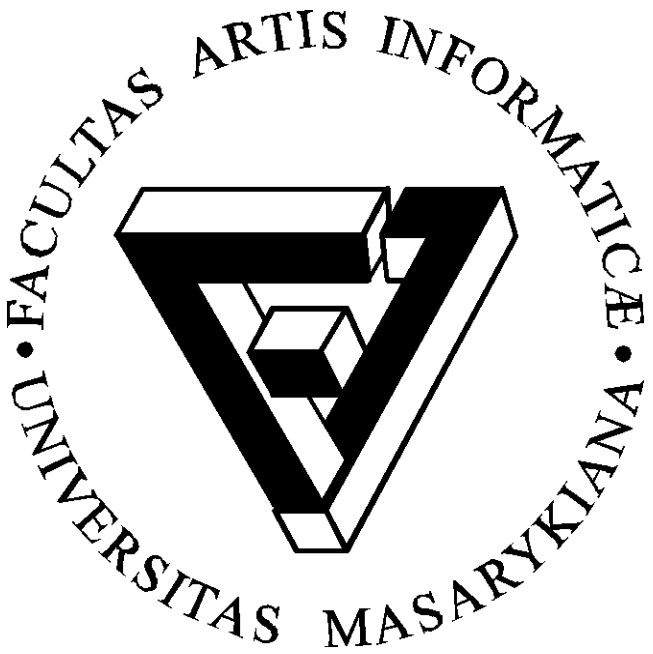



# Úvod do systému Maple

$$\frac{x^3+x}{x^2-1} = x + \frac{1}{x-1} + \frac{1}{x+1}$$

**Jiří Hřebíček & Jan Kohout**

 Brno 2004

# Obsah

# Úvod

## Historie vzniku této knihy

V této kapitole bude stručně uvedena historie vzniku této publikace, která vznikla rámci řešení projektu Fondu rozvoje vysokých škol (FRVŠ) v okruhu F1 č. 499/2004. Nejprve shrneme využívání systémů počítačové algebry na vysokých školách v České republice, dále na Masarykově universitě v Brně, jmenovitě pak na Fakultě informatiky.

## Systémy počítačové algebry na vysokých školách

Charakteristickým rysem inovace výuky matematiky, fyziky, chemie, ekonomie, biologie, atd. ve studijních programech universit v České republice, Evropské unie a v zemích OECD se v posledních deseti letech stává používání nových informačních a komunikačních technologií (ICT) v rámci budování nového vědního oboru „Computational Science“ (Gander & Hřebíček, 2004) . Jedná se například o využití:

- systémů symbolických výpočtů (Symbolic Computing Systems, SCS), resp. systémů počítačové algebry (Computer Algebra Systems, CAS), (http://www.symbolicnet.org/toc.html), kde mezi nejvýznamnější systémy v této oblasti patří například: Axiom (http://axiom.axiom-developer.org/) podporovaný CAISS (Center for Algorithms and Interactive Scientific Software, City College, New York), Derive od firmy Texas Instruments (http://www.derive.com) , Maple od Maplesoft Inc. (http://www.maplesoft.com), MathCAD od MathSoft Inc. (http://www.mathsoft.-com), Mathematica od Wolfram Reasearch, Inc. (http://www.wolfram.com), MuPAD (http://research.mupad.de/) vyvíjený universitou Paderhorne a firmou SciFace Software GmbH (http://www.sciface.com/), atd.
- systémů pro matematické modelování a vědecké výpočty, např. pro řešení složitých diferenciálních a integrálních úloh metodou konečných prvků, hraničních prvků, kde mezi nejvýznamnější programové balíky, patří např. ANSYS, ADYNA, Matlab, apod., dále systémů pro statistické vyhodnocování dat kde mezi nejvýznamnější systémy patří např. SPSS, SAS, Statgraphics, apod., databázových systémů kde mezi nejvýznamnější systémy řízení databází patří např. Oracle, Progress, apod.
- informačních zdrojů na Internetu, ze kterých je možno „stáhnout“ nejen odborné publikace a články, ale i řešení modelových problémů pomocí příslušné ICT.

V České republice jsou významné aktivity českých universit v této oblasti podporovány jak ze strany Ministerstva školství mládeže a tělovýchovy například formou udělování grantů z FRVŠ (Dočkal&Doupovec, 2000), (Dalík et all., 2001), atd. nebo i z prostředků Grantové agentury ČR. V souvislosti s těmito skutečnostmi vznikla významná poptávka po ICT ze strany univerzit na celém světě včetně univerzit z České republiky. Tradičně v čele brněnskými a pražskými vysokými školami jako např. Masarykova universita v Brně (MU), Vysoké učení technické v Brně (VUT), České vysoké učení technické v Praze (ČVUT), Karlova universita v Praze (KU) a Vysoká škola chemicko technologická (VŠCHT). Tyto a další vysoké školy již dnes ve výuce na různých stupních široce využívají ICT v rámci multilicencí jednotlivých licencí.

Budeme to dokumentovat na příkladu využívání systému počítačové algebry Maple, který se v současné době využívá v České a Slovenské republice na sedmnácti vysokých školách včetně VUT (Dočkal&Doupovec, 2000) a MU (Plch, 1998), (Došlá, Plch, Sojka, 1999) v Brně, které začaly Maple systematicky využívat ve výuce matematiky již v roce 1994 jako jedny z prvních v ČR. Maple dále ve výuce využívají KU, ČVUT (http://math.feld.cvut.cz/nemecek/maple/knihovny9/knihovny.html) (Němeček, 2002, 2004) a VŠCHT (http://www.vscht.cz/mat/p1/pas/pas_maple.html) v Praze, Mendlova zemědělská a lesnická universita (MZLU) a Universita obrany (UO) v Brně (Hřebíček, 2002). Dále se Maple

využívá ve výuce matematiky a vědeckých výpočtů na katedrách matematiky na Západočeské univerzitě v Plzni, Jihočeské univerzitě v Českých Budějovicích (Hašek, Klufová, Rost, 2002), (Klufová, Hašek 2002), Technické univerzitě v Libereci, Ostravské univerzitě v Ostravě, Univerzitě Palackého v Olomouci, univerzitě T. Bati ve Zlíně a Slezské univerzitě Opavě, dále Slovenské technické university v Bratislavě, Technické univerzitě v Košicích, atd. Nové možnosti využití Maple ve výuce přináší jeho nové vlastnosti využitelné pro potřeby e-learningu (Hřebíček et all, 2004, 2004a, 2004b)

Kromě vysokých škol využívá Maple i celá řada středních škol, gymnázií a vyšších odborných škol. Pro vědecké výpočty a výzkumné účely je Maple využíván na deseti ústavech AV ČR (např. Fyzikální ústav, Matematický ústav, Ústav teorie informace, Ústav termomechaniky, atd.) a AV SR.

V průběhu roku 2001 vznikly první elektronické učební texty věnované systému Maple verze 7 ve spolupráci Ústavu matematiky a deskriptivní geometrie fakulty stavební VUT v Brně (UMDG FAST) a Katedry informačních technologií Fakulty informatiky MU v Brně (KIT FI). Bylo to v rámci řešení společného projektu FRVŠ 0083/2001. Pro výuku matematiky byly v Maple verze 7 vytvořeny následující elektronické učební texty ve formě mapleovských zápisníků, které jsou uloženy na serveru UMDG FAST:

- *Úvod do systému Maple 7* (http://math.fce.vutbr.cz/vyuka/matematika/uvod_do_maple/uvod-maple.pdf), obsahující: Symbolické výpočty, charakteristika systému Maple 7, uživatelské pracovní prostředí, práce se zápisníky, základní příkazy a operace v Maple 7, Interaktivní nápověda, grafika, knihovny, práce s Internetem.
- *Diferenciální počet v Maple 7* (http://math.fce.vutbr.cz/vyuka/matematika/diferencialni_pocet/), obsahující: Způsoby definice funkcí jedné proměnné, výpočet hodnot, konstrukci složených funkcí a kreslení grafů v Maple. Vysvětlení pojmu limity, určení intervalů spojitosti a bodů nespojitosti funkcí, způsoby vyšetřování vlastností funkcí v okolí daného bodu, prostředky pro výpočet limity ve vlastních i nevlastních bodech. Prostředky Maple pro stanovení definičních oborů funkcí. Definici derivace, pravidla pro jejich výpočet, příklady výpočtu derivací, konstrukce tečen ke grafům, vyšetřování průběhu funkcí a výpočet Taylorových polynomů.
- *Integrální počet v Maple 7t* (http://math.fce.vutbr.cz/vyuka/matematika/integralni_pocet/), věnovaný tématům: Primitivní funkce, neurčitý integrál – úvod, integrace per partes a integrace substitucí pro neurčitý integrál, integrace racionálních funkcí. Určitý integrál – úvod, integrace per partes a substituční metoda pro určitý integrál, příklady na procvičení, užití integrálního počtu, nevlastní integrál**.**

- *Diferenciální rovnice 2. a vyšších řádů v Maple 7 – řešené příklady* (**http://math.fce.vutbr.cz/vyuka/matematika/diferencialni_rovnice/) ,** jehož obsahem jsou kapitoly věnované metodám řešení homogenních rovnic, nehomogenních rovnic metodou variace konstant, nehomogenních rovnic metodou neurčitých koeficientů. V těchto kapitolách jsou výpočtové postupy pro řešení uvedených typů rovnic analyzovány krok za krokem. Do závěrečné kapitoly jsou pak vloženy procedury, které rovnice uvedených typů řeší automaticky.

Tyto elektronické učební texty byly vytvořeny autorským kolektivem Ing. J. Hřebíčkové, RNDr. J. Slaběňákové z UMDG FAST a Prof. RNDr. J. Hřebíčka, CSc., Mgr. J. Pešla, Mgr. J. Ráčka z KIT FI v jazyku HTML a odtud byly hyperlinkovými odkazy volány aktivní výpočtové části jako samostatné zápisníky systému Maple verze 7.

## Systémy počítačové algebry na FI MU

Informatika jako vědní obor je nezastupitelná v oblasti systémů počítačové algebry. Ačkoliv jsou tyto systémy a zejména Maple určeny pro řešení matematických problémů, jejich součástí je netriviální programovací jazyk, jehož znalost by měla být základní dovedností každého absolventa bakalářského nebo magisterského studia odborné informatiky (pro případ studenta zaměřeného do oblasti vědeckých výpočtů životní nutností), (Hřebíček, Pitner, Buchar, 1997). Systém Maple se rovněž využívá v různých oborech (matematika, fyzika, biologie) na Přírodovědecké fakulty (Plch, 1998), (Plch, Došlá, Sojka, 1999), (Došlá, Plch, Sojka, 2002) a Ekonomicko správní fakulty MU v Brně. Na rozdíl od těchto fakult je ovšem cílem Fakulty informatiky naučit studenty využívat Maple nejen pro konkrétní vědecké výpočty, ale i k naprogramování daných postupů a algoritmů, které pak mohou být dále využity jak ve výuce, tak ve výzkumu.

Proto v minulých letech v minulých letech vznikla na FI MU v Brně řada publikací (Hřebíček, Ráček, 2001, 2001a), (Hřebíček, Pešl, Ráček, 2002) a bakalářských (Koudelák 2003, Soldánová 2003, Rosický 2002) a diplomových prací (Cikalo 2004), které se věnovaly popisu základních funkcí a využívání systému Maple jak ve výuce, tak i ve vědeckých výpočtech.

Denní magisterské a bakalářské studium programu „Informatika“ je na FI MU v Brně organizováno kreditním systémem, který umožňuje značnou flexibilitu. Na druhé straně to však znamená, že je poměrně omezen počet a rozsah povinných předmětů, mezi které by mělo patřit alespoň základní seznámení se systémy počítačové algebry. Tato publikace přináší inovovaný učební text základní kurz předmětu „Systémy počítačové algebry“ na FIMU v Brně:

Stávající předmět „Systémy počítačové algebry“ tak mohl být upraven tak, aby reagoval na současný vývoj CAS ve světě s tím, že je zaměřen na systém Maple, který je na MU v Brně dlouhodobě licencován. Maple se v současné době zaměřuje na nové využití ICT, tj. konektivitu systémů a spolupráci s Internetem, prezentaci výsledků výpočtů na WWW, síťových služeb a usnadnění práce budoucím uživatelům s využitím balíků pro tvorbu Javovských appletů. Nezanedbatelná je i snaha tvůrců Maple o podporu programů spolupracujících interaktivně se studentem na matematickém výpočtu, čímž systém výrazně napomáhá k pochopení látky z oblasti matematiky.

Výuková náplň předmětu „Systémy počítačové algebry“, která se zaměřovala na vysvětlení základních sad příkazů pro výpočty ze všech relevantních oblastí matematiky a na použití nejjednodušších programátorských technik, je rozšířena o tvorbu Mapletů, interaktivních výpočetních programů a webových prezentací výsledků. S tím souvisí i zařazení kapitoly o řídících strukturách (cyklech, podmínkách apod.), které mají naučit studenta nejen používat existující příkazy Maplu, ale také naprogramovat vlastní aplikace pro řešení specifických matematických úloh.

# Kapitola 1

## Základní popis systému Maple

V této kapitole se naučíte, jak využívat ICT, speciálně systém Maple pro symbolické výpočty. Seznámíte se podrobně s jeho verzí 9.5 a kdykoli v textu budeme mít tuto verzi na mysli budeme psát Maple 9.5.

Dříve než se budeme věnovat problematice využití ICT pro symbolické výpočty, tak si připomeňme, jak je chápeme v současné době. Výpočty na počítačích souvisejí se slovesem počítat užívaným v kontextu nového vědního oboru „Computational Science“ (http://www.siam.org/cse/report.htm). Sloveso počítat (anglicky compute) bývá obvykle užíváno ve významu počítat s čísly, tedy provádět s nimi základní aritmetické operace sčítání, odečítání, násobení a dělení, tj. numerické výpočty. Numerický výpočet však v současné době neznamená pouze základní aritmetické operace, ale i mnohem složitější výpočty jako např. stanovení řešení soustav rovnic, numerických hodnot matematických funkcí, nalezení kořenů polynomů a vlastních čísel a vektorů matice, apod.. Základem numerických výpočtů v počítačích je, že aritmetické operace jsou vyhodnocovány nejprve obecně a teprve následně číselně. Mimoto tyto numerické výpočty nejsou ve většině případů přesné, protože se téměř vždy jedná o čísla zapsaná v počítači v pohyblivé řádové čárce, kde mantisa má omezený počet cifer. V minulých padesáti letech se však numerické výpočty na počítačích rozšířily do té míry, že pro většinu uživatelů počítačů znamenají matematické výpočty na počítačích a numerické výpočty totéž, což však v již není v posledních letech pravda v souvislosti s rozvojem vědního oboru „Computational Science“.

Do tohoto vědeckého oboru patří např. vědecké výpočty a matematické modelování („Scientific Computing“), jejich další důležitou část, *symbolické a algebraické výpočty* v další části této kapitoly podrobněji vysvětlíme.

## 1.1 Symbolické výpočty

Stručně můžeme symbolické a algebraické výpočty charakterizovat jako výpočty se symboly reprezentujícími matematické objekty. Tyto symboly mohou reprezentovat jednak čísla (celá, racionální, reálná a komplexní, případně i algebraická), booleovské hodnoty (pravda, nepravda, nevím) a znaky (písmena abecedy a další symboly), jednak mohou být používány pro matematické objekty (proměnné, matematické výrazy, rovnice a identity, posloupnosti, množiny, vektory, matice, polynomy a funkce jedné a více proměnných a jejich derivace a integrály, grafy funkcí jedné a dvou proměnných a jejich animace, systémy rovnic, nerovnic a algebraické struktury jako grupy, okruhy a algebry a jejich prvky, orientované grafy, apod.), datové struktury (tabulky a datové soubory) a vykonavatelé procedury (funkce, grafy, algoritmy, atd.).

Kromě toho přídavné jméno *symbolický* zdůrazňuje, že konečným cílem řešení matematického problému je vyjádření jeho řešení v explicitním analytickém tvaru nebo nalezení jeho symbolické aproximace (např. konečné funkční řady). Pojmem *algebraický* myslíme, že výpočty jsou prováděny přesně v souladu s pravidly algebry, namísto použití přibližné aritmetiky v pohyblivé řádové čárce, tak jak je tomu u klasických numerických výpočtů.

Příklady symbolických a algebraických výpočtů jsou například zjednodušování a úpravy matematických výrazů, analytické řešení rovnic, rozklad polynomů, derivování, integrování funkcí a rozvoj funkcí v řady, analytické řešení obyčejných i parciálních diferenciálních a integrálních rovnic, exaktní řešení systémů rovnic i nerovností atd.

V posledních padesáti letech byl v matematice udělán velký pokrok v oblasti teoretických základů symbolických a algebraických výpočtů a algoritmů, k jejich rozvoji došlo jak na základě využití informačních technologií a prováděním matematických výpočtů a modelování na počítačích, tak využitím komunikačních technologií v počítačových sitích, zejména pak využití Internetu a intranetu. To vedlo celosvětově ke vzniku nového oboru, který je označován mnoha různými jmény: *symbolické a algebraické výpočty, systémy symbolických výpočtů, operace se symboly, operace s výrazy*, *počítačová algebra* atd. Bohužel termín symbolický výpočet je užíván v mnoha odlišných kontextech, jako logické programování a umělá inteligence, tak má velmi málo společného s matematickými výpočty.

Abychom se vyhnuli mylnému překladu, budeme pro tento druh výpočtů na počítačích dále užívat anglickou zkratku *CAS (Computer Algebra System),* tj. česky *„systémy počítačové algebry",* i když budou v zahraničním odborném tisku někdy značeny anglickou zkratku *SCS* nebo *SAC (Symbolic Computation Systems* nebo *Symbolic and Algebraic Computation)*, tj. česky *„systémy symbolických výpočtů“* nebo *„symbolické a algebraické výpočty“*.

Přehled těchto systémů počítačové algebry je možno nalézt na webu na adrese http://www.SymbolicNet.org/systems/ , nebudeme je zde rozvádět, neboť byly uvedeny v úvodní kapitole.

Ačkoli mnoho standardních algebraických operací s matematickými výrazy je možno provádět jen s papírem a tužkou, tak u rozsáhlejších symbolických výpočtů začíná být nevýhodou větší délka vzorců a tím zdlouhavější práce matematika, který výrazy upravuje. Další nevýhodou těchto operací je nutné neustále maximální soustředění, aby se dosáhlo bezchybnosti algebraických operací a tím správnosti výsledku.

Při ilustraci CAS se omezíme v dalším na systém Maple a nejnovější verzi 9.5 stručně popíšeme.

## 1.2 Charakteristika systému Maple 9.5

Maple je programový systém počítačové algebry vyvinutý během uplynulých dvaceti pěti let společně na několika západních universitách, přičemž největší podíl práce vykonala skupina vědců sdružená pod názvem "Symbolic Computation Group" na universitě ve Waterloo v Kanadě a dále pak na federální technické universitě ETH Zürich ve Švýcarsku, kam část této skupiny přešla v roce 1990. V současné době je Maple komercializován a jeho další vývoj řídí kanadská firma Maplesoft Inc., (http://www.maplesoft.com) sídlící ve Waterloo ve státě Ontario.

Jméno Maple by mohlo být odvozeno z anglického akronyma *Mathematics pleasure (Matematika potěšením),* neboť Maple je skutečně příjemným prostředím pro využívání matematiky na počítači. Během posledních deseti let se Maple stal jedním z nejmodernějších a nejintenzivněji se rozvíjejících systémů počítačové algebry ve světě.

Současná verze 9.5 systému Maple (zkráceně Maple 9.5) umožňuje provádět jak symbolické a numerické výpočty a vytvářet grafy, tak doplňovat je vlastními texty a vytvářet tak tzv. *hypertextové zápisníky* (anglicky "worksheet"). Takto vytvořené zápisníky umožňuje Maple 9.5 ukládat do souboru na počítači ve svém speciálním maplovském formátu MW, který je uložen ve formátu XML. Soubory ve formátu MW umožňuje Maple 9.5 načítat zpět ke zpracování, což umožňuje snadnou přenositelnost maplovských zápisníků mezi nejrůznějšími počítačovými platformami a operačními systémy.

Soubory lze také volitelně exportovat do formátu Latexu, HTML, RTF a nově i MathML, což je rozšíření HTML pro prezentaci matematických textů na webu. Maple 9.5 dále

umožňuje automatický převod svých příkazů a procedur do programovacích jazyků C, Fortran 77 a Java.

V Maplu 9.5 se používá vlastní programovací jazyk čtvrté generace podobný Pascalu s mnoha předdefinovanými funkcemi a procedurami. Maplovské funkce pokrývají mnoho odvětví matematiky od základů diferenciálního a integrálního počtu, lineární algebry, řešení rovnic, až k řešení diferenciálních a diferenčních rovnic, diferenciální geometrii a logice.

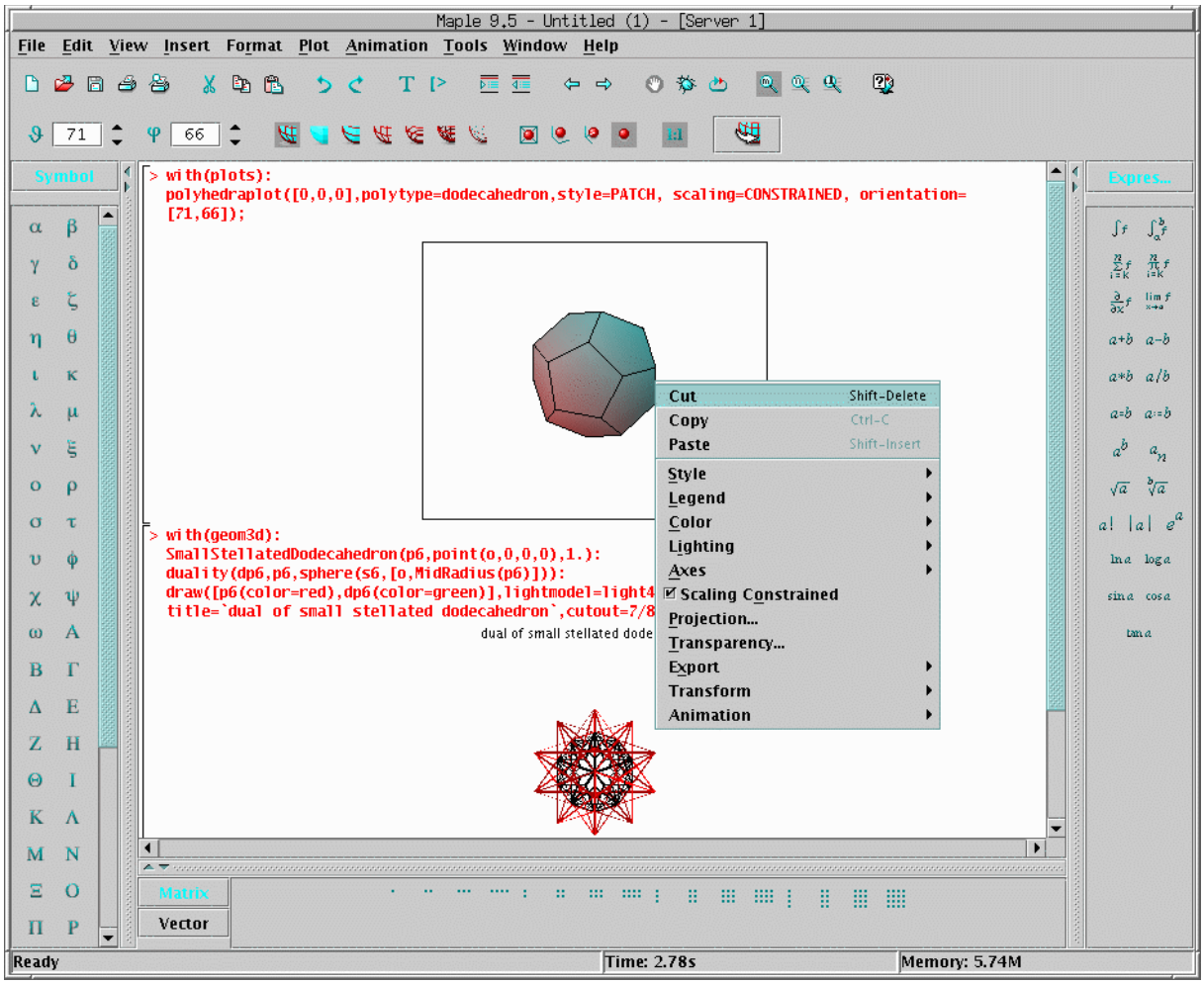
## 1.3 Uživatelské pracovní prostředí

V této sekci je popisováno uživatelské pracovní prostředí programu Maple 9.5, které odpovídá pracovním prostředím v operačních systémech Windows NT/2000/XP a Linux.

Systém počítačové algebry Maple 9.5 používá grafické uživatelské prostředí. Klíčovou částí v rámci tohoto prostředí je pracovní a komunikační rozhraní označované jako ***zápisník***.

V rámci okna programu Maple 9.5 se standardně nachází hlavní nabídka menu a ovládací lišta reprezentovaná řadou ikon, viz obr 1.1. Pod ovládací lištou je kontextově závislá pracovní lišta s nabídkou aktuálně přístupných funkcí. Na levé straně je okno s takzvanými „paletami“ umožňujícím jednoduché a rychlé vkládání výrazů, symbolů řecké abecedy, matic a vektorů. Toto okno lze libovolně dělit a přesouvat i na všechny ostatní strany plochy, jak je zjevné z obr. 1.1.

Hlavní část okna programu Maple 9.5 je vyhrazena pro pracovní plochu, na které je umístěn právě zpracovávaný zápisník, při spodním okraji okna programu je stavový řádek informující o časové a paměťové náročnosti právě zpracovávaného zápisníku.




*Obrázek č. 1.1: Okno programu Maple 9.5 s dvěma otevřenými zápisníky*

Nabídka menu Maple 9.5 obsahuje vedle běžných menu pro práci se soubory (File, Edit, View, Window, Help) též některé kontextově závislé položky, jako například Style, Color, Axes, Projection, s aktuální nabídkou akcí a parametrů funkcí, použitelných v rámci aktivního objektu v zápisníku. Nejdůležitější z těchto funkcí jsou dostupné pomocí ikon na kontextově závislé pracovní liště. Většinu takto interaktivně nabízených parametrů funkcí (jako například způsob zobrazení souřadných os, barvu a tloušťku čar používaných v grafech atd.) je však možno definovat přímo jako parametry maplovských funkcí a příkazů použitých v zápisníku.

Na ovládací liště Maple 9.5 jsou dostupné běžné ikony usnadňující práci se zpracovávanými zápisníky, jako například otvírání, ukládání a tisk souborů zápisníku, práce se schránkou, funkce „zpět“ (undo), přepínání mezi zapisováním do zápisníku formou prostého textu nebo formou aktivních maplovských příkazů, zastavení aktuálního výpočtu (na méně výkonných počítačích velice užitečná funkce), lupa a některé další.

Kontextově závislá pracovní lišta odvíjí svoji podobu od právě aktivního objektu v zápisníku. Na obr. 1.1 je vidět pracovní lišta s tlačítky vztahujícími se k 3D grafu funkce. Konkrétně nabízí otáčení 3D objektu kolem středu ve dvou směrech, přepínání mezi různými způsoby znázornění plochy, od vybarvené sítě až po tečkovanou interpretaci, různé způsoby znázornění souřadných os až po vyžádání konstantního měřítka na všech osách. Měřítka jednotlivých os automaticky přizpůsobují tvaru a rozsahu grafu.

Pro jednoduché interaktivní vkládání a editaci matematických výrazů jsou v levé liště umístěny navíc čtyři takzvané *palety* nazvané Výrazy, Symboly, Matice a Vektory. Umístění

těchto menu lze změnit a umístit je nahoru, vlevo, vpravo nebo dolů, přičemž jednotlivé palety lze rozmístit na různé strany nebo případně zcela vypnout.

Práce s paletami je jednoduchá. Stačí kliknout na ikonu reprezentující naši volbu, a do zápisníku je na aktuální pozici vložen výraz, který již stačí pouze naplnit. Hodnoty, které je potřeba naplnit jsou přitom zvýrazněny. V prostředí Windows navíc lze ikonu „chytit“ myší a umístit přímo na správnou pozici.

Pro úplné začátečníky, kteří se neorientují ve významu jednotlivých ikon, je tu možnost zapnout si interaktivní nápovědu, tzv. „balloon help“, automaticky vypisující názvy a stručnou charakteristiku funkcí spojených s danou ikonou.

## 1.4 Práce se zápisníky

Zápisníky jsou hlavním uživatelským pracovním prostředím pro ovládání Maple 9.5. Umožňují uživateli pohodlné zadávání prováděcích příkazů, zároveň též slouží k okamžité prezentaci výstupů systému Maple 9.5. Po spuštění programu Maple 9.5 se na jeho pracovní ploše automaticky otevře nový prázdný zápisník.

Práce v novém zápisníku spočívá v zapisování vstupních příkazů Maple 9.5 do maplovské vstupní oblasti. Tyto příkazy jsou uvozeny symbolem „>“ (větší než) a zobrazují se červeně. Ukončují se buď středníkem „;“ nebo dvojtečkou „:“. Může se jich zapsat více na jeden řádek. Chceme-li je zapsat do samostatných řádků, tak po jejich ukončení musíme stisknout současně klávesy `[Shift + Enter]`.

Po stisknutí klávesy `[Enter]` se všechny příkazy z maplovské vstupní oblasti provedou. Pokud je příkaz ukončen středníkem jeho výsledky se zobrazí modře v „standardní matematické notaci“, je-li však ukončen dvojtečkou, tak se jeho výsledky nezobrazí.

Množina vstupních oblastí s jejich odpovídajícími výstupy se v zápisníku Maplu nazývá prováděcí skupina (anglicky *execution group*). Zápisník dále může obsahovat samostatné textové oblasti v matematické notaci (anglicky *paragraphs*) a hypertextové odkazy (anglicky *hyperlinks*) a tabulkové kalkulátory (anglicky *spreadsheet*). Pro zpřehlednění lze zápisník rozdělit do sekcí (anglicky *sections*) a podsekcí.

### 1.4.1 Prováděcí skupiny

Prováděcí skupiny usnadňují práci s matematickým jádrem systému Maple 9.5. Umožňují přehledné zadávání a provádění jednotlivých příkazů a následné zobrazování výsledků.

Prováděcí skupiny tvoří základní výpočetní bloky v zápisníku. Jejich primárním účelem je kombinování jednoho či více příkazů a jejich numerických, symbolických nebo grafických výsledků do samostatné, znovupoužitelné jednotky. Příkazy a výsledky patřící do jedné prováděcí skupiny lze snadno poznat díky veliké hranaté svorce nalevo od vstupních řádku. První vstupní příkaz je uvozen symbolem „>“ (větší než). Pokud máme umístěn kurzor v prováděcí skupině, tak stisknutím klávesy `[Enter]` se provedou všechny příkazy ve skupině.

**■ Příklad 1.1:**

Příklad prováděcí skupiny se třemi příkazy na jednom a na třech řádcích s jejich zobrazenými výsledky.

```
> r:=8; Obvod:= evalf(2*Pi*r); Obsah:= evalf(Pi*r^2);
```

$$r := 8$$

$$Obvod := 50.26548246$$

$$Obsah := 201.0619299$$

```
> r:=8;
  Obvod:= evalf(2*Pi*r);
  Obsah:= evalf(Pi*r^2);
```

$$r := 8$$

$$Obvod := 50.26548246$$

$$Obsah := 201.0619299$$

Výsledky příkazů mohou být *numerické* (Příklad 1.1), *symbolické* (Příklad 1.2) či *grafické* (viz Příklad 1.3).

**■ Příklad 1.2:**

Text se před příkaz vkládá do prováděcí skupiny pomocí menu `Insert/Paragraph` nebo pomocí současného stisknutí kláves `[Ctrl+Shift+K]`. Ukážeme to v následující prováděcí skupině:

Výpočet třetí mocniny součtu $a+b$

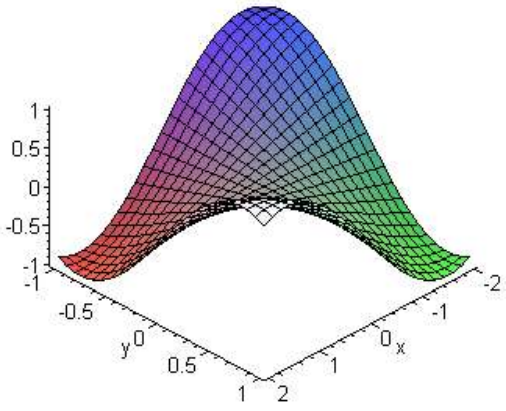
```
> expand((a+b)^3);
```

$$a^3+3a^2b+3ab^2+b^3$$

**■ Příklad 1.3:**

Následující příkaz vykreslí třírozměrný graf funkce *sin(xy)*. v prováděcí skupině. S tímto grafem lze pomocí ukazatele myši otáčet kolem středu.

```
> plot3d(sin(x*y) , x = -2..2, y = -1..1 );
```




Z důvodů přehlednosti v dalších příkladech budeme již vynechávat velkou hranatou svorku na levé straně, označující příkazy a jejich výsledky, patřící do jedné prováděcí skupiny. Nebude-li uvedeno jinak, budeme automaticky předpokládat, že příkazy stojí samostatně, tedy že nejsou sdruženy v žádných prováděcích skupinách.

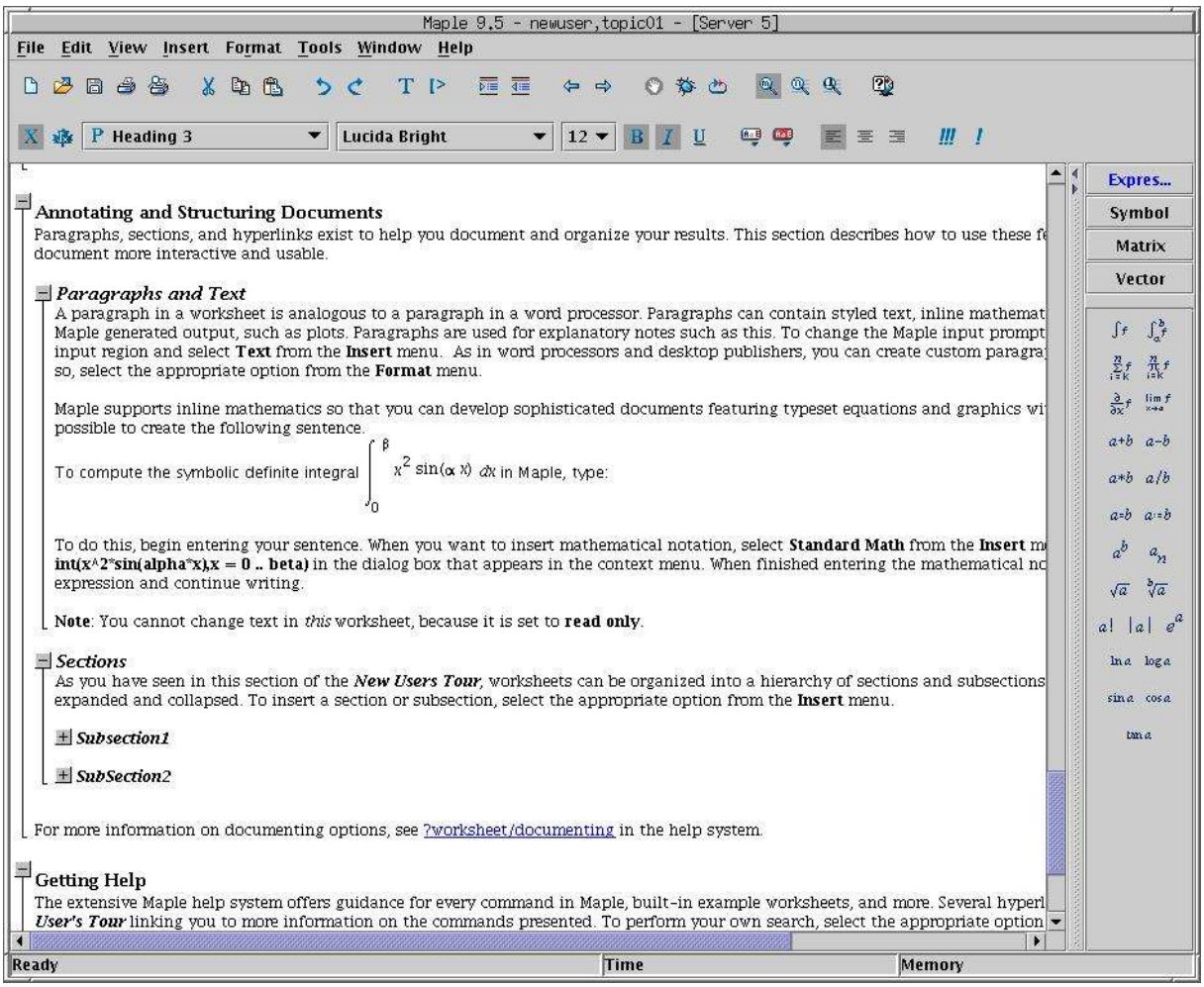
### 1.4.2 Textové odstavce

Odstavec prostého textu v programu Maple 9.5 je stejný jako text, se kterým se lze běžně setkat v textových procesorech. Odstavce mohou obsahovat text formátovaný pomocí různých stylů. Text může být obsažen i v jednotlivých prováděcích skupinách. Funguje zde i běžné zarovnávání textu na střed či okraje stránky stejně jako u textových procesorů.Všechny běžně využívané funkce pro formátování textu jsou snadno dostupné na kontextově závislé pracovní liště. Maple umožňuje vkládat i text matematické notaci pomocí menu **Insert** a funkce **Standard Math**.

### 1.4.3 Sekce v zápisnících

Zápisník programu Maple 9.5 se tedy skládá z prováděcích skupin, doprovodných textů, hypertextových odkazů a tabulkových kalkulátorů. Tyto součásti zápisníku mohou být organizovány v hierarchické struktuře založené na *sekcích* a *podsekcích*.

Zápisník lze pomocí *sekcí* přehledně členit. Sekce jsou části zápisníku, které lze podle potřeby jednotlivě otevírat a zase naopak zavírat.




*Obrázek č. 1.2: Zápisník Maple 9.5 popisující anglicky strukturování zápisníků*

K ovládání sekce slouží tlačítko vlevo od názvu sekce se symboly "+" respektive "-".

- "+" značí, že sekce je momentálně zavřená a kliknutím na tlačítko se otevře
- "-" znamená, že sekce je otevřená a po kliknutím na tlačítko se uzavře

Veškerý obsah konkrétní otevřené sekce je pro přehlednost a snazší orientaci v textu ohraničen svorkou na levé straně, podobně jako prováděcí skupiny.

Díky tomu, že podsekce lze do sebe libovolně vnořovat, jsou výborným nástrojem určeným k víceúrovňovému strukturování zápisníku. V sekcích lze například skrýt podrobný vysvětlující text, který při běžném provádění zápisníku ruší a znepřehledňuje výsledky, nebo dokonce i alternativní způsob řešení daného problému zapsaný v prováděcích skupinách.

Sekce a podsekce se vkládají do zápisníku pomocí menu `Insert/Section`, respektive `Insert/Subsection`.

### 1.4.4 Hypertextové odkazy

Do textu zápisníku lze vkládat i hypertextové odkazy, které v rámci Maple 9.5 rozlišujeme na dva druhy.

- *odkazy na zápisníky,*
  které slouží k přímému propojení jednotlivých zápisníků do ucelené struktury, po které se pak můžeme snadno pohybovat. Můžeme takto zpřístupnit i zápisníky umístěné v síti Internet a Intranet.

- *odkazy na soubory jiného typu,*
  které slouží k tomu, aby při vybrání takovéhoto odkazu je spuštěn externí prohlížeč webovských stránek, kterému je předán vybraný odkaz. Takovéto odkazy pracují buď s URL adresou daného souboru nebo je nutno jim zadat absolutní cestou v rámci lokální adresářové struktury. Bohužel při předání odkazu externímu prohlížeči dochází často k nežádoucí interpretaci odkazu. z toho důvodu nelze u těchto odkazů používat směřování pomocí relativních cest, což poněkud snižuje možnosti jejich využití.

### 1.4.5 Průběh výpočtu ve vytvořeném zápisníku

Způsob práce a prohlížení již vytvořených zápisníků se odvíjí od způsobu jejich sestavení, tj. od uspořádání jejich prováděcích skupin. Pokud se umístí kurzor na libovolný řádek v prováděcí skupině a stiskne klávesa `[Enter]`, znamená to, že se všechny příkazy v dané prováděcí skupině provedou, a to v pořadí, v jakém jsou ve skupině uvedeny za sebou. Výsledky výpočtu se však zobrazí na konci prováděcí skupiny. Kurzor se poté automaticky přesune na první řádek následující prováděcí skupiny.

Prohlédnout již vytvořený zápisník lze tedy nejlépe takto:

- pomocí myši či klávesnice umístíte kurzor na první řádek první prováděcí skupiny v daném zápisníku a pak stisknete `[Enter]`,
- Maple 9.5 zobrazí výstupy a výsledky vykonané prováděcí skupiny,
- po prostudování výstupů pokračujte dále opět stiskem klávesy `[Enter]`, čímž se spustí a provede následující prováděcí skupina.

Tímto způsobem postupně projdete všechny prováděcí skupiny v zápisníku.

*Poznámka:* Mnohdy se v prováděcích skupinách na konci zápisníku pracuje s dílčími výsledky získanými během výpočtu v předchozích prováděcích skupinách. Pokud příslušné prováděcí skupiny nebyly vykonány, nejsou tyto mezivýsledky k dispozici a po spuštění prováděcí skupiny uvnitř zápisníku se může objevit chybové hlášení. Prováděcí skupiny je tedy třeba volat (vykonávat) v závislosti na daném algoritmu. Obvykle popořadě tak jak jsou v zápisníku postupně definovány.

## 1.5 Komunikace s využitím Internetu

Ve verzi 9.5 došlo k významnému rozšíření spolupráce Maplu s Internetem. Systém umožňuje jak exportování zápisníků do formátu zobrazitelného internetovým prohlížečem, tak i načtení souboru přímo z URL adresy na Inernetu.

Pro načtení souboru z Internetu použijeme nabídku `OpenURL z` menu `File.` Zadáme cestu a jedná-li se o Maplovský soubor, je tento otevřen přímo v Maplu, jinak se spustí internetový prohlížeč (nebo jiný program – volbu spouštěného programu zadáváme při prvním použití této nabídky).

Pro prezentaci zápisníku na webu můžeme použít některou z široké nabídky exportů v menu. Většina z nich je dostupná pomocí nabídky `ExportAs z` menu `File.` Zápisník můžeme takto převést do jednoho z formátů HTML, MathML, LaTeX, MapleText, PlainText nebo RTF. Pro prezentaci na webu jsou nejvýznamnější převody do HTML nebo MathML. Nevýhodou exportu do HTML je, že veškeré výstupy Maplu jsou převáděny jako vložené obrázky v grafickém formátu GIF, naopak jeho výhodou je zobrazitelnost výsledku kterýmkoliv prohlížečem podporujícím grafiku. Nový standard MathML se snaží nedostatky odstranit a matematika je zde prezentována pomocí speciálního rozšíření jazyka HTML, který ovšem nebyl zatím do většiny prohlížečů zabudován a produkt se tudíž zobrazí nekorektně. Tento standard se po zabudování do prohlížečů zřejmě stane hlavním nástrojem pro matematiku na webu, ale dnes je nutné zobrazovat jej speciálním prohlížečem.

Balík MathML ve spojení s balíkem XMLTools umožňuje převést i jednotlivé výrazy (i importovat zpět), jak ukáže následující příklad:

```
> MathML[Export]( a + 2 * b );
```

```
"<math xmlns='http://www.w3.org/1998/Math/MathML'><semantics><mrow xref='i \
   d5'><mi xref='id1'>a</mi><mo>+</mo><mrow xref='id4'><mn xref='id2'>2</m \
   n><mo>&InvisibleTimes;</mo><mi xref='id3'>b</mi></mrow></mrow><annot \
   ation-xml encoding='MathML-Content'><apply id='id5'><plus/><ci id='id1'>a</ \
   ci><apply id='id4'><times/><cn id='id2' type='integer'>2</cn><ci id='id3'>b</ci \

   ></apply></apply></annotation-xml><annotation encoding='Maple'>a+2*b</an \
   notation></semantics></math>"
```

```
> XMLTools[Print]( % );
<math xmlns='http://www.w3.org/1998/Math/MathML'>
  <semantics>
    <mrow xref='id5'
      <mi xref='id1'>a</mi>
      <mo>+</mo>
      <mrow xref='id4'>
        <mn xref='id2'>2</mn>
        <mo>&InvisibleTimes;</mo>
        <mi xref='id3'>b</mi>
      </mrow>
    </mrow>
    <annotation-xml encoding='MathML-Content'>
      <apply id='id5'>
        <plus/>
        <ci id='id1'>a</ci>
        <apply id='id4'>
```

```
          <times/>
          <cn id='id2' type='integer'>2</cn>
          <ci id='id3'>b</ci>
        </apply>
      </apply>
    </annotation-xml>
    <annotation encoding='Maple'>a+2*b</annotation>
  </semantics>
</math>
```

```
> MathML[Import]( % );
```

$$a + 2\,b$$

Kromě zápisníků nám Maple umožňuje exportovat i jednotlivé grafy. Klepnutím pravým tlačítkem na obrázek v zapisníku Maplu se objeví roleta, která mezi nabídkami opět obsahuje volbu `ExportAs`. Zde můžeme volit mezi grafickými formáty EPS, GIF, JPG, BMP nebo WMF. Pro převod zápisníku Maple 9.5 do LaTeXu se samozřejmě používá volba EPS, naopak pro převod zápisníku Maple 9.5 do webové prezentace se volí formát GIF nebo JPG (jak již bylo zmíněno, při konverzi do HTML dojde ke zobrazení nejen obrázků, ale i matematických vzorců ve formátu GIF).

Pokud nám tato nabídka nepostačuje, lze před převodem nastavit argumenty obrázku jako výšku, šířku či vzhled stránky. Toto demonstrujeme na příkladu nastavení postscriptu:

```
plotsetup (ps, plotoptions=`colour=cmyk, width=4in,
height=3in, leftmargin=3cm, bottommargin=2cm, noborder`);
```

## 1.6 Základní příkazy a operace

Syntaxe používaná systémem Maple 9.5 pro zapisování příkazů je podobná syntaxi programovacích jazyků Pascal a C. Příklady použité v této kapitole i všechny ostatní příklady jsou zobrazeny tak, jak se zpravidla jeví při skutečné práci s Maplem 9.5.

### 1.6.1 Způsob zápisu příkazů v zápisnících

Nejprve uvedeme několik obecných informací o způsobu zapisování příkazů na řádky zápisníku.

- Každý příkaz v zápisníku, obsahující příkaz Maple 9.5, musí být ukončen středníkem „ ; “ nebo dvojtečkou „ : “.
- Po stisknutí klávesy `[Enter]` je celá aktuální prováděcí skupina předána jako vstup "výpočetnímu jádru" programu Maple 9.5, které ji zpracuje.

Pokud je řádek se zpracovávaným mapleovským příkazem zakončen středníkem, znamená to, že výsledek provedené operace se zobrazí na dalším řádku. Je-li řádek zakončen dvojtečkou, Maple 9.5 vyhodnotí zadaný příkaz, ale výsledek nezobrazí a očekává další příkaz. Tohoto se využívá zejména k potlačení tisku mezivýsledků v průběhu delších výpočtů.

Při zápisu posloupnosti příkazů či dlouhých algebraických výrazů je zapotřebí mít možnost přecházet na další řádek bez toho, že by se prozatím napsaný příkaz nějakým způsobem vyhodnocoval, jak se v zápisníku děje po prostém stisku klávesy `[Enter]`. To je možné provádět následujícím způsobem:

- Posunout kurzor na nový řádek, bez vyhodnocení dosud napsané části příkazu, nám v zápisnících umožňuje kombinace kláves `[Shift] + [Enter]`.

### 1.6.2 Aritmetické operace a přiřazení hodnoty proměnné

V Maple9.5 v aritmetických operacích pracuje se dvěma typy jmen: indexovaným jménem a neindexovaným jménem. Tato jména musí splňovat následující podmínky:

- musí začínat písmenem, ať už malým či velkým (POZOR! Maple 9.5 rozlišuje mezi malými a velkými písmeny)
- mohou být složeno z následujících znaků:
  - písmena,
  - číslice,
  - znaku podtržítko „ _ “ ,
- maximální délka jména je závislá na počítačové platformě, na 32-bitových systémech (případ Windows 9x) je to 524 271 znaků, na systémech 64-bitových je maximální délka jména proměnné 34 359 738 335 znaků.
- jako jméno nemůžeme definovat žádné z klíčových slov, která uvedeme dále v odstavci 1.6.3..

*Poznámka:* Znak podtržítko „_“ na začátku jména proměnné je vyhrazen pro globální systémové proměnné, proto se doporučuje taková jména nepoužívat. Jména obsahující lomítko (/) jsou obecně rezervována pro kódy v knihovnách Maple a neměla by se používat pokud to není explicitně uvedeno v nápovědě. Jména, která končí vlnovkou (~) označují v Maple proměnné předepsanými předpoklady (např.jejich hodnota je kladná, záporná,atd.) a rovněž by neměla být používána.

Každé jméno a obecně i výraz má v Maple přiřazen typ. Těchto typů je velké množství a nebudeme je zde všechny probírat. U výrazu se jménem `vyraz` zjistíme jeho typ pomocí příkazu `whattype(vyraz)`, který nám dá jako výsledek, že se jedná o:

- Aritmetický, relační nebo logický operátor(`` `*` ``, `` `+` ``, `` `.` ``, `` `..` ``, `` `::` ``, `` `<` ``, `` `<=` ``, `` `<>` ``, `` `=` ``, `` `^` ``, `` `and` ``, `` `not` ``, `` `or` ``),
- Celé, racionální, reálné číslo v pohyblivé řádové čárce nebo komplexní číslo (`integer, fraction, float, complex`),
- Strukturu typu pole, matice, sloupcový nebo řádkový vektor, tabulka, možina, seznam, textový řetězec (`array, Matrix, Vector[column], Vector [row],table,set, list, string`),
- Funkci, proceduru, posloupnostní výraz nebo nekonečnou řadu (`function, procedure, exprseq, series`),
- Nevyhodnotitelný výraz (`uneval`).

V Maple pracujeme v aritmetických operacích s čísly, která jsou definována následujícím způsobem:

- *Celá čísla* jsou v Maplu representována libovolně dlouhými sekvencemi jedné nebo více desítkových číslic a je jim přiřazen typ `integer`. Maximální délka celého čísla (počet jeho číslic) závisí na operačním systému a můžeme ji zjistit pomocí příkazu `kernelopts(maxdigits)`.
- *Racionální čísla* (zlomky) jsou reprezentovány v Maple ve tvaru zlomku „celé číslo se znaménkem/přirozené číslo“, přičemž všichni společní činitelé jsou ve

zlomku zkráceni. Je jim přiřazen typ `fraction`. Podobně jako celá čísla mají i racionální čísla v Maple libovolnou délku.

- *Reálná čísla* v Maplu jsou „softwarově“ representována v pohyblivé řádové čárce, tj. dvojicí celých čísel: mantisou - *M*, exponentem – *E* a mají hodnotu $M*10^E$. Je jim přiřazen typu `float`. Počet číslic, které mantisa zobrazuje, je určen v systémové proměnné `Digits` (standardně je rovna 10). Přítomnost desetinného čísla v aritmetickém výrazu obecně značí, že výraz se dále bude vyhodnocovat jako reálné číslo. Zřejmé to bude v dalších příkladech.
- *Komplexní číslo* může mít v Maplu dva tvary - samotné imaginární číslo nebo obecné komplexní číslo. Samotná imaginární čísla jsou tvaru `I*y`, kde `y` je celé, racionální nebo reálné číslo a `I` je imaginární číslo *i* (tj. $\sqrt{-1}$). Obecná komplexní čísla jsou tvaru `x+I*y`, kde `x` a `y` jsou v Maple representovány jako celé, racionální nebo reálná čísla nebo výrazy. Pokud jeden z výrazů `x` nebo `y` v komplexním čísle je typu `float` (v pohyblivé řádové čárce), potom jsou oba výrazy automaticky převedeny na reálný typ do pohyblivé řádové čárky. Komplexním číslům je přiřazen typ `complex`.

Následující příklad ukazuje, jak pracovat v Maple 9.5 s přiřazením hodnoty maplovské proměnné v oboru celých, racionálních, reálných a komplexních čísel a běžnými aritmetickými operacemi, jejichž výsledky zobrazuje Maple přesně v oboru celých a racionálních čísel (zlomků).

■ **Příklad 1.4:**

```
> a:= 5/2;
```

$$a := \frac{5}{2}$$

```
> b:=6!;
```

$$b := 720$$

```
> c:= a*b;
```

$$c := 1800$$

```
> d:= c^2+Pi*(a-e);
```

$$d := 3240000 + \pi\left(\frac{5}{2} - e\right)$$

```
> e:= c-12345;
```

$$e := -10545$$

```
> f:=d*(b-e^a);
```

$$f := \left(3240000 + \frac{21095}{2}\pi\right)(720 - 111197025\sqrt{-10545})$$

```
> g:=2*b/100-1/3.;
```

$$g := 14.06666667$$

```
> komplexni_cislo:=d*(a+I*b);
```

$$komplexni\_cislo := \left(\frac{5}{2} + 720\,I\right)\left(3240000 + \frac{21095}{2}\pi\right)$$

```
> E:=exp(1);
```

$$E := e$$

Z výše uvedených příkazů je vidět, že Maple 9.5 zobrazuje výsledky v „matematické notaci“ a pokud možno přesně, tj. v tvaru celých čísel, zlomků, případně výrazů, kde se vyskytující matematické konstanty (zde `Pi` znamená Ludolfovo číslo $\pi$ a $e$ základ přirozených logaritmů) i odmocniny.

Maple má rezervována pro vybrané matematické konstanty následující jména: `Catalan, gamma, infinity, Pi.` Jejich význam je uveden v tab. 1.1. Maple však nemá pro konstantu $e$ (základ přirozených logaritmů) rezervováno žádné jméno, ale její hodnota se spočítá pomocí exponenciální funkce `exp`.

*Tabulka č. 1.1: Základní aritmetické operátory, funkce a konstanty*

| Zápis v Maple 9.5 | Význam | Matematický zápis |
|---|---|---|
| **`x + y`** | Součet | $x + y$ |
| **`x - y`** | Rozdíl | $x - y$ |
| **`x * y`** | Součin | $x * y$ |
| **`x / y`** | Podíl | $\frac{x}{y}$ |
| **x^y** nebo **`x**y`** | Umocňování | $x^y$ |
| **`Sqrt(x)`** nebo **x^(1/2)** | druhá odmocnina | $\sqrt{x}$ |
| **`x!`** | Faktoriál | $x!$ |
| **`Abs(x)`** | absolutní hodnota | $\lvert x \rvert$ |
| **`I`** nebo **`sqrt(-1)`** | imaginární jednotka | $i$ nebo $\sqrt{-1}$ |
| **`Pi`** | Ludolfova konstanta | $\pi$<br>přibližně 3.141592654 |
| **`Catalan`** | Katalánská konstanta | $\sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+1)^2}$<br>přibližně 0.915965594 |
| **`Gamma`** | Eurelova konstanta | $\lim_{n\to\infty} \left( \left( \sum_{i=1}^{n} \frac{1}{i} \right) - \ln(n) \right)$<br>přibližně 0.5772156649 |
| **`Infinity`** | symbol pro nekonečno | $\infty$ |

Pro logické konstanty má Maple, který pracuje s tříhodnotovou logikou, rezervována následující jména: `true` (pravda), `false` (nepravda), `FAIL` (nevím).

Pokud chceme vyčíslit výsledek vyhodnocení výrazu pouze přibližně v pohyblivé řádové čárce, např. jako desetinné číslo, tak použijeme maplovský příkaz `evalf`, který zobrazuje čísla uvedené v jeho prvním parametru v pohyblivé řádové čárce s mantisou standardně na 10 desetinných míst.

```
> evalf(f);
```

$$2.356657883\,10^{9} - 3.737494002\,10^{16}\,I$$

```
> evalf(komplexni_cislo);
```

$$8.182839872\,10^{6} + 2.356657883\,10^{9}\,I$$

V případě, že požadujeme větší počet desetinných míst, uvedeme jejich počet jako další parametr v příkazu `evalf.` Počet míst mantisy není omezen, ale prodloužíme tím délku výpočtu.

```
> evalf(f,20);
```

$$2.356657882929891607 8\,10^{9} - 3.737493999985409998 7\,10^{16}\,I$$

```
> evalf(komplexni_cislo,30);
```

$$8.18283987128434586045671182772\,10^{6} + 2.35665788292989160781153300638\,10^{9}\,I$$

```
> evalf(E,50);
```

$$2.7182818284590452353602874713526624977572470937000$$

Z příkladu 1.4 je vidět, že v prostředí zápisníku můžeme definovat proměnné. Proměnnou definujeme prostým přiřazením hodnoty jménu proměnné nebo jejím zápisem ve výrazu. Přiřazovací operátor má podobu

*„jméno proměnné := “.*

*Poznámka:* (samotný operátor „ = “ má jiný význam).

Následující tab. 1.1 podává stručný přehled aritmetických operátorů, základních matematických funkcí a konstant definovaných v Maple 9.5. Zároveň ukazuje způsob jejich zápisu v prostředí zápisníku.

■ **Příklad 1.5:**
Chceme-li tedy například přiřadit hodnotu 77, proměnné `poloměr` a nezobrazit výsledek zapíšeme v Maplu 9.5 příkaz:

```
> polomer := 77:
```

*Poznámka:* Maple 9.5 na platformách s Windows 9x sice podporuje kódování češtiny, nicméně její používání ve jménech proměnných rozhodně nedoporučujeme, neboť to může vést k neočekávané interpretaci zadaných jmen.

Vzhledem k tomu, že jména proměnných, jakož i funkcí a procedur, jsou v Maplu 9.5 rozlišována podle malých a velkých písmen, tak po zadání následujících příkazů obdržíme:

```
> Polomer := 50;
```

$$Polomer := 50$$

```
> polomer-Polomer;
```

$$27$$

Proměnné může být přiřazen jakýkoli výraz či jiná maplovská struktura. Jako příklad uveďme tři následující přiřazení:

■ **Příklad 1.6:**

Přiřazení rovnice:

```
> rovnice := 3 * x^2 - x + 11 = 0;
```

$$rovnice := 3\,x^{2} - x + 11 = 0$$

Přiřazení struktury definující graf funkce *2*cos(x):*

```
> graf := plot (2*cos(x), x = -Pi .. Pi):
```

Strukturu definující graf funkce zde z úsporných důvodů nevypisujeme.

Pro úplnost ještě uveďme, že v Maple 9.5 je možno definovat jako jméno proměnné dokonce posloupnost znaků obsahující mezery. Takové jméno proměnné se uzavře do jednoduchých uvozovek a dále se s ním pracuje běžným způsobem.

```
> `Toto je jmeno promenne` := 23;
```

$$Toto\ je\ jmeno\ promenne := 23$$

*Poznámka:* Při výpisu výsledků však Maple 9.5 jednoduché uvozovky uzavírající jméno proměnné vynechává. Výstupy z prováděcích skupin, ve kterých jsou použita jména proměnných s mezerami, jsou pak zpravidla velice těžce čitelné, proto je nedoporučujeme používat. Uzavřením jména proměnné do jednoduchých uvozovek se význam jména nemění.

### 1.6.3 Chráněná jména v Maple 9.5 :

Chráněná jména v Maple 9.5 jsou jména maplovských příkazů, funkcí, operátorů a struktur. Nelze je obecně používat jako jména v zápisnících Maple 9.5. Existují jich stovky a jsou zařazena do typu `protect`. Tento typ slouží jako ochrana vyhrazených jmen před náhodným předefinováním jejich významu.

Patří sem jména matematických funkcí jako `sin`, `cos`, `ln`, `exp`, atd., dále výše uvedených matematických konstant `false`, `gamma`, `infinity`, `true`, `Catalan`, `FAIL`, `Pi` a mnohá další. Jména typu `protect` nemohou být použita jako jména proměnných, při pokusu přiřadit jim nějakou hodnotu akce skončí chybovým hlášením:

```
> Pi := 1;
Error, attempting to assign to `Pi` which is protected
```

V případě potřeby můžeme jakékoli chráněné jméno typu `protect` z tohoto typu vyjmout a poté předefinovat. K tomu slouží příkaz `unprotect`.

```
> unprotect(Pi): Pi := 1;
```

$$\pi := 1$$

A naopak, jméno lze zařadit do typu `protect`, takže v průběhu další práce s Maple 9.5 je zaručeno, že jeho hodnota nebude náhodně či omylem změněna. K tomuto slouží mapleovský příkaz `protect`, který ilustrujeme dále. Nejprve příkazem `restart` nastavíme v Maple 9.5 původní hodnoty globálních maplovských konstant včetně `Pi`.

```
> restart: alpha:=Pi/6;
```

$$\alpha := \frac{1}{6}\pi$$

```
> protect('alpha'):
> alpha:=Pi;
Error, attempting to assign to `alpha` which is protected
```

Seznam všech jmen Maple9.5 zařazených v typu `protect` obdržíme po zadání příkazu:

```
> select(type,{unames(),anames(anything)},protected):
```

Vzhledem k jejich počtu je zde nebudeme všechny vypisovat,uvedeme jen ty nejvýznamnější. Mezi ně patří například: `.`, `_X`, `_Z`, `@`, `@@`, `int`, `ln`, `O`, `Pi`, `add`, `if`, `mul`, `seq`, `*`, `**`, `||`, `^`, `..`, `=`, `>`, `>=`, `Im`, `<`, `<=`, `<>`, `+`, `Re`, `abs`, `and`, `cat`, `{}`, `$`, `gc`, `has`, `lhs`, `map`, `max`, `min`, `not`, `op`, `or`, `?()`, `?[]`, `rhs`, `[]`, `xor`, `::`, `!`, `odd`, `set`, `Fraction`, `infinity`, `overflow`, `undefined`, `underflow`, `assigned`, `piecewise`, `algebraic`, `anything`, `constant`, `equation`, `fraction`, `function`,

identical, imaginary, indexable, negative, Matrix, RootOf, Vector, matrix, trace, vector, ARRAY, Catalan, Complex, FAIL, Float, Integer, TABLE, false, gamma, package, restart, true, eval, evalf, evalhf, evaln, time, inner, Array, DEBUG, ERROR, array, bind, coeff, coeffs, convert, degree, denom, diff, divide, nops, normal, numer, order, print, quit, readlib, remove, rtable, savelib, select, series, sign, sort, stop, table, taylor, tcoeff, trunc, type, union, atomic, boolean, complex, even, finite, float, indexed, integer, list, literal, module, name, negint, negzero, nonreal, numeric, polynom, posint, poszero, positive, procedure, protected, rational, realcons, relation, sequential, verboseproc, radical, range, ratpoly, sfloat, string, symbol, tabular, uneval, zppoly.

### 1.6.4 Odkazování na předchozí výsledky

Každý příkaz a funkce zadané a zpracované systémem Maple 9.5 v rámci zápisníku má svoji hodnotu, kterou získá během svého vyhodnocení. Na hodnotu již vyhodnocených příkazů se můžeme přímo odkazovat pomocí speciální systémové proměnné „ % “. To znamená, že není vždy nutné přiřazovat výsledek příkazu do nějaké proměnné. To je velmi užitečné, zajímá-li nás z celého průběhu výpočtu, rozloženého do více kroků, pouze konečný výsledek.

U složitějších úloh bychom však určitě nevystačili pouze s odkazem na bezprostředně předcházející výsledek. Maple 9.5 však umožňuje odkazovat se zpětně až na třetí předchozí vyhodnocený výsledek.

Symbol: % - odkazuje na poslední vyhodnocený výsledek
%% - odkazuje na předposlední vyhodnocený výsledek
%%% - odkazuje na třetí vyhodnocený výsledek ve zpětném pořadí

**■ Příklad 1.7:**

Využití symbolu % ilustrujme na následujících příkazech.

```
> (5 + 2)^2;
                                  49
> % - 45;
                                   4
> 1 + %%;
                                  50
> (%%% / 7) * 3;
                                  21
> sqrt(%%%);
                                   2
```

Z výše uvedeného příkladu je vidět, že zjednodušení práce může být výrazné. Přesto doporučujeme přiřadit výsledek příkazu do proměnné pro lepší přehled v zápisníku.

## 1.7 Jednoduchá nápověda k maplovskému příkazu

Velmi důležitým pomocníkem ve využívání Maplu je jeho nápověda. Tato interaktivní nápověda v Maple 9.5 slouží ke snadné a rychlé orientaci ve tisících maplovských příkazů, funkcí, knihoven a balíků a jejich parametrů. V Maple 9.5 je nápověda řešena jako systém textových dokumentů propojených hypertextovými odkazy. Každá standardní funkce (příkaz)

Maple 9.5 má zpracovánu vlastní stránku s nápovědou. Jednotlivé stránky nápovědy k maplovskému příkazu mají pevnou strukturu, skládající se z následujících po sobě jdoucích částí:

- název a charakteristika příkazu
- popis volání příkazu
- definice parametrů příkazu
- podrobný popis vlastností příkazu
- příklady použití příkazu
- seznam hypertextových odkazů na příbuzná témata v nápovědě Maple 9.5

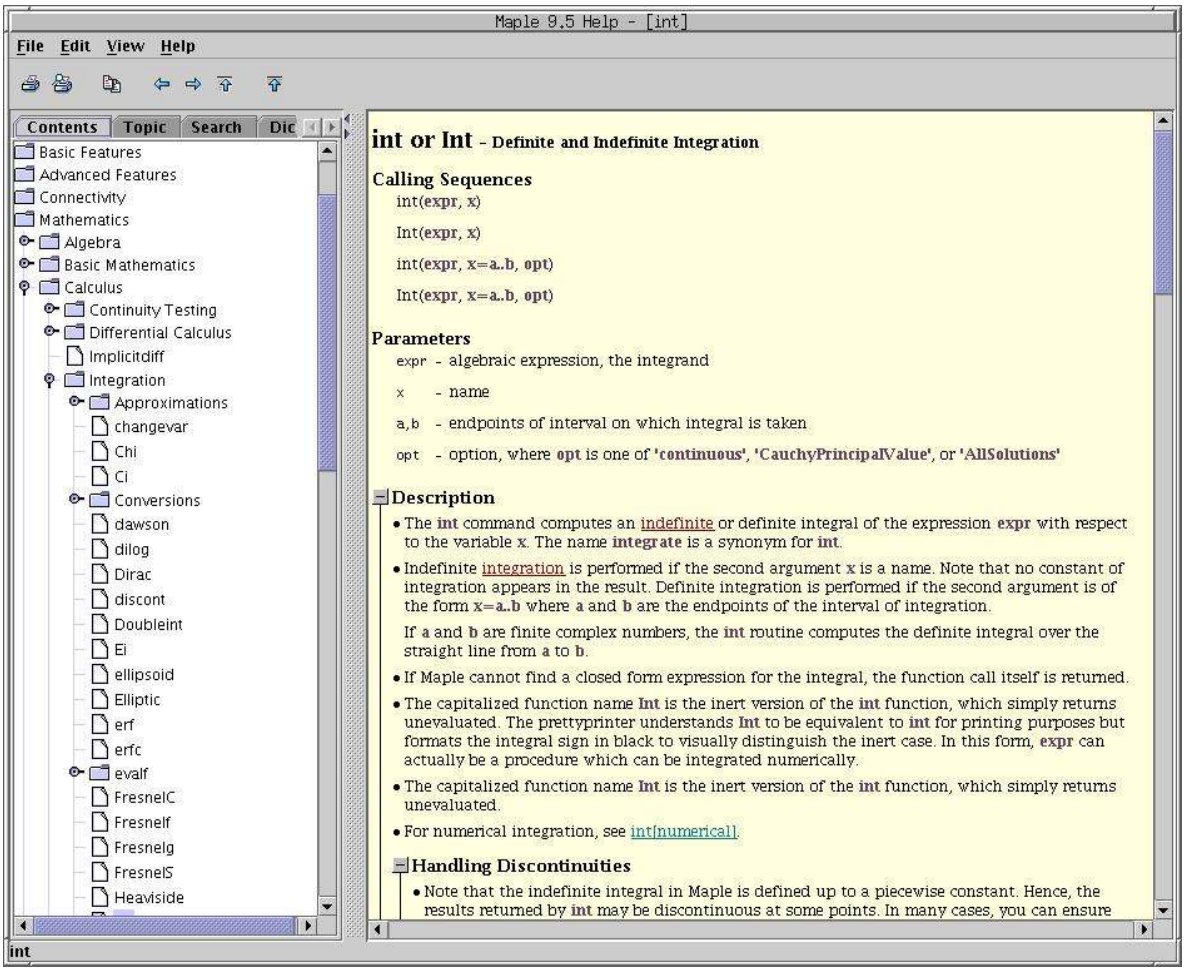
Systém interaktivní nápovědy v Maple 9.5 spočívá v tom, že umožňuje zpřístupnit přímo stránku týkající se zadané funkce. Ukážeme to na příkladu.

**■ Příklad 1.8:**

Potřebujeme-li například nápovědu k příkazu `int`, napišme na řádek zápisníku příkaz

```
> ?int
```

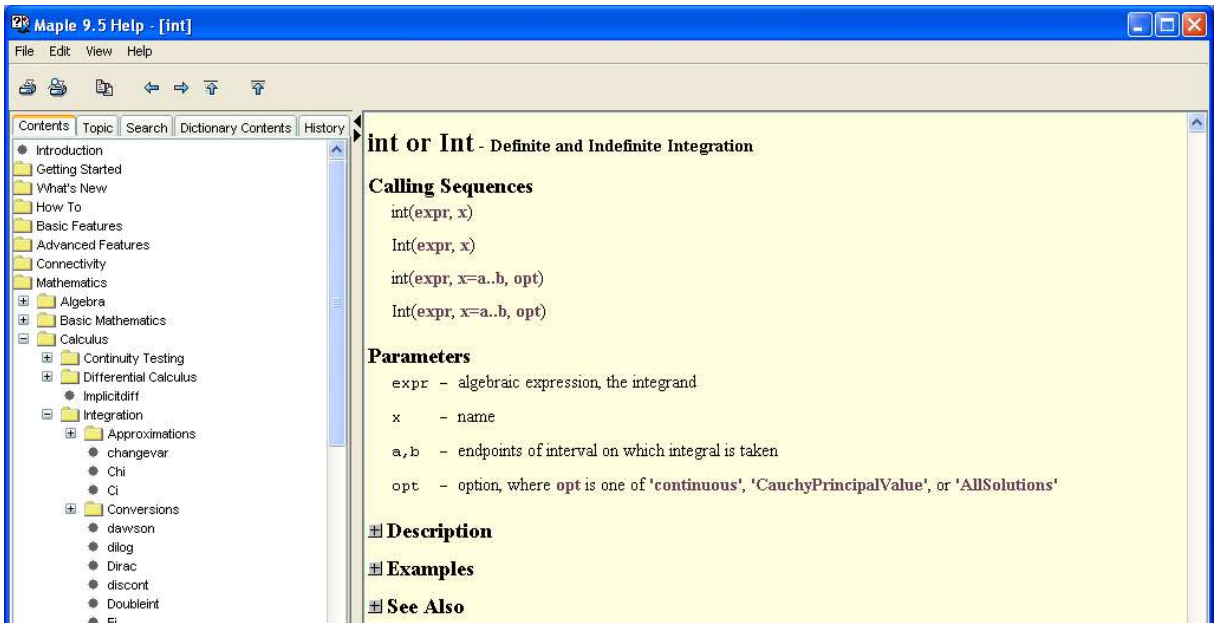
a stiskněme klávesu [Enter] . Tím se zobrazí v novém okně požadovaná stránka nápovědy.




*Obrázek č. 1.3: Nápověda k příkazu* `int`

Vzhledem k tomu, že nápověda bývá u některých příkazů velmi rozsáhlá, tak je možno si ji zobrazit ve „sbaleném“ tvaru pomocí příkazu

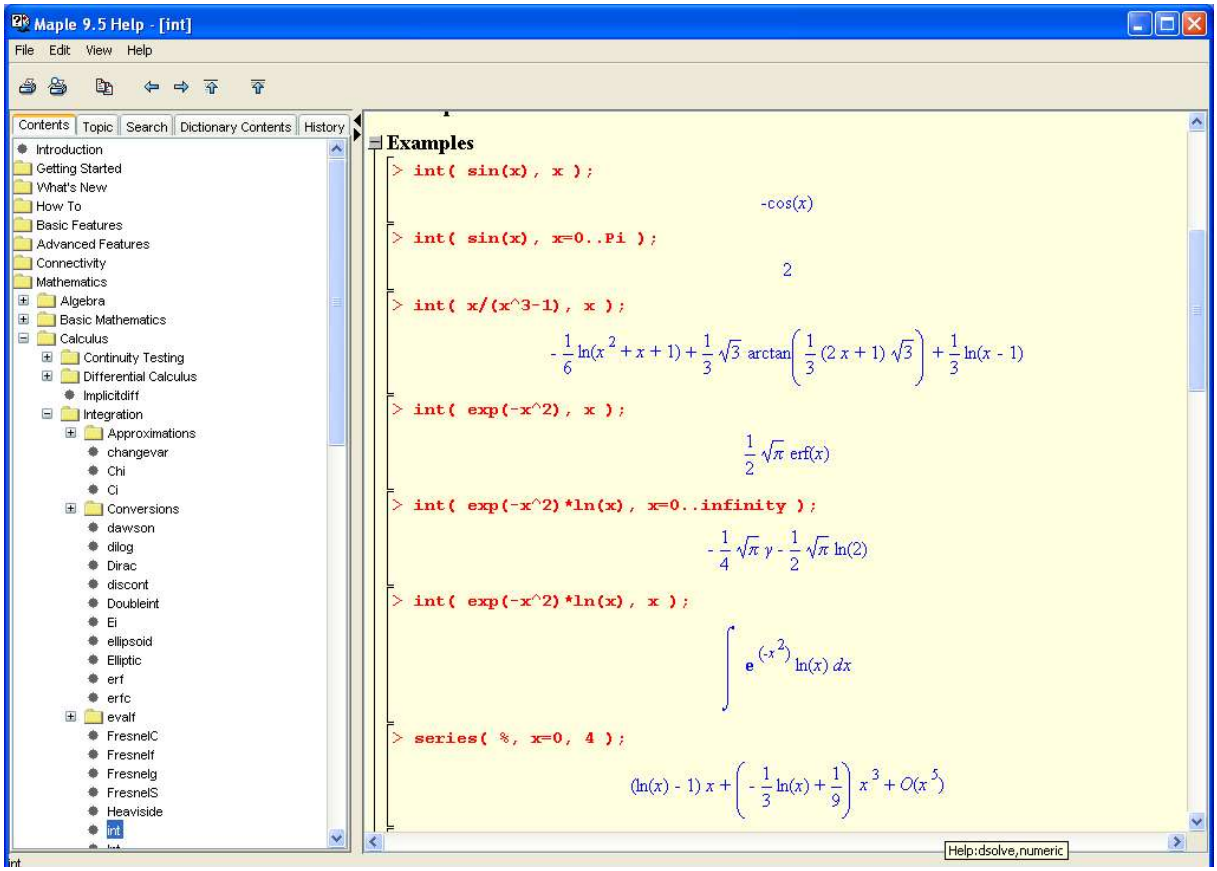
```
> ??int
```




*Obrázek č. 1.4: „Sbalená“ nápověda k příkazu* `int`

Někdy nám stačí se podívat jen příklady použití daného příkazu `int` uvedené v nápovědě. Pak stačí zapsat příkaz

```
> ???int
```




*Obrázek č. 1.5: Příklady z nápovědy k příkazu* `int`

Je-li již jméno funkce `int` v zápisníku použito, tak stačí na něj přenést kurzor, myší či klávesovými šipkami, a poté vybrat položku `` `Help on int` `` z menu `Help` zápisníku nebo jen stisknout klávesy [`Ctrl + F1`].

Pokud jsme v situaci, kdy hledáte funkci, jejíž jméno neznáte, nezbývá vám, než procházet systém stránek nápovědy manuálně. Systém stránek nápovědy je však hierarchicky členěn a díky tomu se v něm dá velice dobře vyhledávat. Otevřít systém nápovědy můžeme například pomocí položky `Using Help` v menu `Help` .

Podrobnějšímu objasnění práce s nápovědou je věnována samostatná kapitola.

## 1.8 Knihovny funkcí

Maple 9.5 při řešení úloh umožňuje použít obrovské množství příkazů a matematických funkcí. Ve standardní instalaci současné verze Maple 9.5 je dostupných více než 3000 funkcí. Funkce jsou uloženy v takzvané *knihovně funkcí*.

Z důvodu zpřehlednění práce se stovkami přístupných funkcí jsou funkce v rámci knihovny rozděleny do takzvaných *balíků* (packages).

Kromě standardní knihovny funkcí je možno s programem Maple 9.5 využívat též tzv. s*hare library*. Tato knihovna je sestavena z funkcí a balíků napsaných uživateli Maple 9.5 a obsahuje mnoho funkcí použitelných přímo v praxi. Tato knihovna je nyní uložena na webovském serveru Maple na adrese http://www.mapleapps.com v tzv. „Maple Application Center“.

### 1.8.1 Standardní knihovní funkce – Standard library functions

Funkce (příkazy) z tohoto balíku jsou automaticky přístupné v Maple a je možné je volat jejich jménem ihned po spuštění Maple. Není nutné je inicializovat jako funkce z ostatních balíků, viz 1.8.2. Sem patří hlavně běžné matematické funkce ale také příkazy Maple 9.5, umožňující manipulaci a vyhodnocování zpracovávaných výrazů. Seznam standardních knihovních funkcí je dostupný příkazem:

```
> ?index[function];
```

### 1.8.2 Knihovní balíky – Library packages

Knihovní balíky jsou určité množiny příkazů, v rámci dané knihovny příkazů systému Maple 9.5. Většinou jsou to funkce (příkazy) vztahující se k řešení určité třídy matematických úloh, které jsou společně zařazeny do jedné knihovny balíku příkazů. Potřebujeme-li například pracovat s objekty třírozměrného euklidovského prostoru, tak všechny příkazy vztahující se k této problematice jsou zařazeny do jednoho balíku, konkrétně se jedná o balík `geom3d`.

Pokud chceme používat příkazy z některého balíku, existují dva způsoby jejich volání:
a) volání pomocí *dlouhých jmen* (`long names`)
b) volání pomocí *krátkých jmen* (`short names`)

***Volání pomocí dlouhých jmen***

Dlouhá jména příkazů není třeba inicializovat. Dlouhé jméno příkazu je určeno *jménem balíku,* do kterého je příkaz zařazen a *jménem příkazu* v rámci balíku.

Chceme-li například použít příkaz `point` z balíku `geom3d`, pak jej voláme následovně: `geom3d[point]`. Tento způsob volání vylučuje kolizi jmen příkazů z různých balíků.

Možnost nechtěného předefinování významu některého příkazu během výpočtu může být velmi nebezpečné. Uvědomíme-li si navíc, že v programu Maple 9.5 můžeme používat tisíce různých příkazů, vidíme, že opatření zabraňující takovéto chybě jsou zcela nezbytná.

***Volání pomocí krátkých jmen***

Výše uvedenému volání příkazů pomocí dlouhých jmen se můžeme vyhnout pomocí *inicializace zkráceného volání příkazů*. Tato možnost slouží hlavně k zjednodušení práce Maplu a usnadňuje též manipulaci s příkazy.

K inicializaci používání krátkých jmen příkazů z daného balíku slouží příkaz `with`, jehož parametrem je jméno balíku. Tímto příkazem inicializujeme současně všechny příkazy z balíku. Pokud jako druhý parametr uvedeme jméno konkrétního příkazu, tak se inicializuje pouze uvedený příkaz z balíku, např. příkazem `with(plots,animate)` se inicializuje příkaz `animate` z balíku `plots`.

Při inicializaci určitého knihovního balíku pomocí příkazu `with` jsou na výstupu vypsána všechna jména jeho nově inicializovaných příkazů. Pokud došlo ke shodě jmen některého inicializovaného příkazu s již definovaným jménem příkazu či proměnné, platí význam nově inicializovaného příkazu!

Ukážeme to na příkladu zkráceného volání příkazů z balíku `plots`:

```
> with(plots);
```

```
[animate, animate3d, animatecurve, arrow, changecoords, complexplot, complexplot3d, conformal,
  conformal3d, contourplot, contourplot3d, coordplot, coordplot3d, cylinderplot, densityplot, display,
  display3d, fieldplot, fieldplot3d, gradplot, gradplot3d, graphplot3d, implicitplot, implicitplot3d, inequal,
  interactive, interactiveparams, listcontplot, listcontplot3d, listdensityplot, listplot, listplot3d, loglogplot,
  logplot, matrixplot, multiple, odeplot, pareto, plotcompare, pointplot, pointplot3d, polarplot, polygonplot,
  polygonplot3d, polyhedra_supported, polyhedraplot, replot, rootlocus, semilogplot, setoptions,
  setoptions3d, spacecurve, sparsematrixplot, sphereplot, surfdata, textplot, textplot3d, tubeplot]
```

Seznam všech dostupných knihovních balíků i s jejich stručným popisem je dostupný pomocí příkazu:

```
> ?index[packages];
```

## 1.9 Matematické funkce

V této části je uveden výčet názvů základních matematických funkcí, jež jsou programem Maple 9.5 podporovány.

### 1.9.1 Trigonometrické a hyperbolické funkce:

Trigonometrické a hyperbolické funkce patří mezi standardní knihovní funkce a nemusí se před svým voláním nijak inicializovat.

| | | | |
|---|---|---|---|
| `sin()` | `arcsin()` | `sinh()` | `arcsinh()` |
| `cos()` | `arccos()` | `cosh()` | `arccosh()` |
| `tan()` | `arctan()` | `tanh()` | `arctanh()` |

`cot()` `arccot()` `coth()` `arccoth()`

### 1.9.2 Logaritmy a exponenciální funkce:

| | |
|---|---|
| `ln(), log()` | přirozený logaritmus, logaritmus o základu **e** |
| `log[b]()` | logaritmus o základu b |
| `log10()` | dekadický logaritmus, je třeba definovat pomocí `readlib(log10);` |
| `exp()` | exponenciální funkce, hodnota exponenciální funkce o základu **e** |

### 1.9.3 Celočíselné funkce:

| | |
|---|---|
| `factorial(),!` | funkce faktoriál, symbol ! se používá v běžné formě postfixové notace |
| `binomial()` | binomický koeficient, volání `binomial(n,k)` kde $0 \le k \le n$ , definován vztahem $\frac{n!}{k!(n-k)!}$ |

# Kapitola 2

# Úpravy matematických výrazů

Při manipulacemi se složitými matematickými výrazy se můžeme snadno dostat do situace, kdy při „ručních“ úpravách těchto výrazů uděláme chybu nebo si nejsme zcela jisti správností dosaženého výsledku. Právě zde se nabízí možnost použití systému Maple s jehož pomocí můžeme algebraický výraz převést do požadovaného tvaru, případně si ověřit již dosažený výsledek. Tím se ušetří velké množství práce a času.

Jednou z velkých předností Maple je jeho schopnost pracovat se symbolickými algebraickými výrazy tak, jak jsme to viděli v předchozí kapitole. Tato vlastnost se možná trochu paradoxně projevuje právě tím, že Maple do vyjádření zadaného algebraického výrazu zasahuje zcela minimálně. Kromě numerických výpočtů číselných aritmetických výrazů a zjednodušování zlomků zde neexistuje prakticky žádné automatické upravování algebraických výrazů do nějakého předepsaného tvaru. Maple ponechávají zcela na uživateli, jakým způsobem bude daný výraz upraven.

V dalším se zaměříme na objasnění základních příkazů pro úpravu matematických výrazů v Maple 9.5, které jsou: `simplify, expand, combine, factor, normal, convert.`

## 2.1 Zjednodušování algebraických výrazů

Pro zjednodušení výrazu se používá nejčastěji příkaz `simplify,` který umožňuje aplikovat celou řadu zjednodušovacích pravidel, které jsou vhodné ke zpracování zadaného algebraického výrazu. Při volání příkazu `simplify` je zadaný algebraický výraz zanalyzován, jsou v něm identifikována volání funkcí, mocniny a odmocniny, a poté jsou provedena všechna zjednodušení platná pro daný výraz.

Pomocí druhého parametru příkazu `simplify` lze specifikovat skupinu zjednodušujících matematických pravidel, jež mají být při zjednodušování výrazu použita. Zjednodušovací pravidla z ostatních skupin pak při tomto zjednodušení použita nebudou. Můžeme tak určit, které matematické funkce v zadaném výrazu zjednodušeny budou a které naopak zjednodušeny nebudou.

Jako druhý parametr funkce `simplify()` lze volit např. tato jména skupin zjednodušujících pravidel:

- `trig` – tento parametr použijeme, pokud chceme ve výrazu zjednodušit obsažené trigonometrické funkce.
- `radical` – parametr se používá pro úpravu výrazů s racionálními mocninami.
- `power` – parametr použijeme v případě, kdy chceme zjednodušit mocniny a logaritmy
- `sqrt` – jedná se o parametr pro úpravu výrazů, v nichž se vyskytuje čtverec mocnin.
- `ln` - parametr je vhodný pro zjednodušování výrazů s logaritmy.
- `@, constant, constants, Ei, GAMMA, Hypergeom, piecewise, polar, RootOf, rtable, siderels, size, wronskian,zero` – tyto parametry jsou méně obvyklé a nebudeme je již podrobněji popisovat, neboť jejich popis můžeme získat pomocí nápovědy.

■ **Příklad 2.1:**

Nejprve ukážeme použití příkazu `simplify` pouze s jedním základním parametrem.

```
> 4^(1/2)+3;
```

$$\sqrt{4}+3$$

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

$$5$$

```
> ((a^4-b^4)/(a^2*b^2))/((1+b^2/a^2)*(1-2*a/b+a^2/(b^2)));
```

$$\frac{a^4 - b^4}{a^2 b^2 \left(1 + \frac{b^2}{a^2}\right)\left(1 - \frac{2a}{b} + \frac{a^2}{b^2}\right)}$$

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

$$\frac{a+b}{-b+a}$$

Jak je vidět, že příkaz `simplify` lze použít jak pro výpočet hodnoty číselného výrazu, tak i pro zjednodušení výrazu s proměnnými. Nyní ukážeme na dvou případech použití příkazu `simplify` s parametrem `trig`:

```
> sin(x)^2+cos(x)^2;
```

$$\sin(x)^2 + \cos(x)^2$$

```
> simplify(%,trig);
```

$$1$$

```
> c:= exp(ln(cos(2*x)+sin(x)^2) + 1);
```

$$c := e^{(\ln(\cos(2x) + \sin(x)^2) + 1)}$$

```
> simplify(c);
```

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

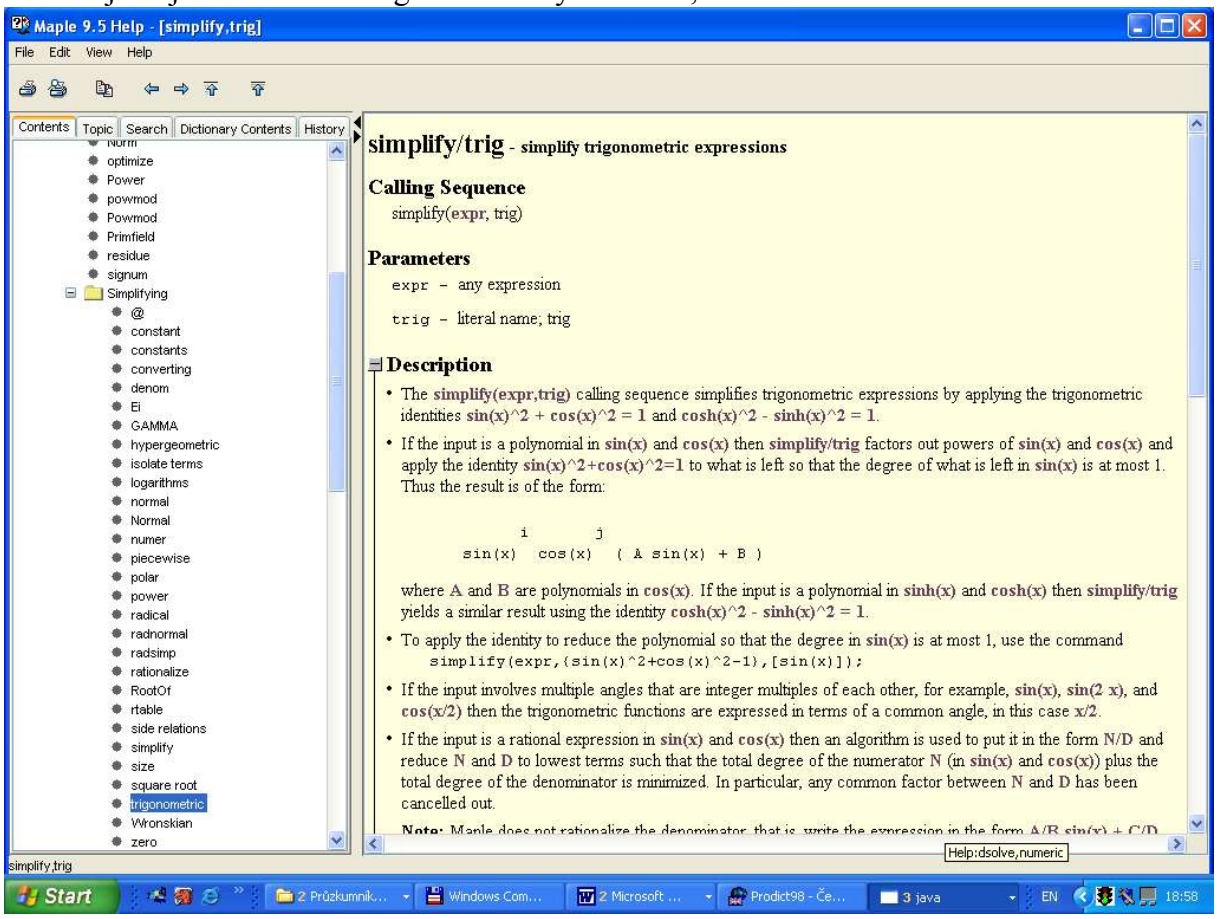
```
> simplify(c,trig);
```

$$e^{(\ln(\cos(x)^2) + 1)}$$

Vidíme, že použití parametru `trig` vede ve druhém případě k složitějšímu výrazu, neboť se nepoužila skupina zjednodušujících pravidel pro funkci `ln`.

*Poznámka:* Pomocí příkazu `?simplify[jméno_skupiny_pravidel]` lze vypsat nápovědu ke každé skupině zjednodušujících pravidel. V nápovědě ke každé ze skupin jsou definována skupinou používaná zjednodušující pravidla. Samotné jméno skupiny vesměs odpovídá typu matematických funkcí, které lze s její pomocí zjednodušovat.

Například pomocí příkazu `?simplify[trig]` se vypíše nápověda pro skupinu umožňující zjednodušování trigonometrických funkcí, viz obr. 2.1:




*Obrázek č. 2.1: Nápověda pro skupinu* `trig`

## 2.2 Rozvoj výrazů

Pro rozvoj výrazů je nejvhodnější příkaz `expand`. Úlohou tohoto příkazu je rozvinout algebraický výraz do řady součtů jeho podvýrazů. Rozvíjí se takové algebraické výrazy, u kterých je to možné, do běžné podoby polynomu. Mimo to příkaz `expand` umí též pracovat s většinou běžných matematických funkcí jako jsou například funkce: *sin, cos, tg, ln, exp,* apod. Dovede také do podoby součtu rozvinout výrazy obsahující zmíněné matematické funkce.

Příkaz `expand` lze volat s jedním nebo více parametry. Prvním a zároveň jediným jeho povinným parametrem je výraz, který má být rozkládán. Jako další parametry lze uvést funkce z výrazu, které nemají být dále rozkládány.

**■ Příklad 2.2:**

V následujících maplovských příkazech jsou ukázány možnosti příkazu `expand` s jedním nebo více parametry:

```
> (x+1)*(x+2)=expand((x+1)*(x+2));
```

$$(x+1)(x+2) = x^2 + 3x + 2$$

```
> sin(x+y)=expand(sin(x+y));
```

$$\sin(x+y) = \sin(x)\cos(y) + \cos(x)\sin(y)$$

```
> a:=(x+1)*(y+z);
```

$$a := (x+1)(y+z)$$

```
> expand(a);
```

$$xy + xz + y + z$$

```
> expand(a,x+1);
```

$$(x+1)y + (x+1)z$$

```
> expand (cos (2*x));
```

$$2\cos(x)^2 - 1$$

```
> expand((1+x)*(1-x)*(x-2)^2);
```

$$-3x^2 - 4x + 4 - x^4 + 4x^3$$

Posledního výraz v podobě polynomu s jeho nesetříděnými mocninami lze setřídit pomocí příkazu `sort()`, a upravit tak pořadí členů polynomu.

```
> sort(%);
```

$$-x^4 + 4x^3 - 3x^2 - 4x + 4$$

S příkazem `expand` jsou úzce spjaty příkazy `expandon`, respektive `expandoff`. Pomocí příkazů `expandon` respektive `expandoff` můžeme předem označit funkce, které mají respektive nemají být rozloženy pomocí příkazu `expand`. Pro názornost a snazší pochopení použití těchto příkazů je uveden následující příklad.

**■ Příklad 2.3:**

Rozdíl mezi použití příkazu `expandon` a `expandoff` ukážeme na příkladu exponenciální funkce:

```
> expand(expandoff()); expandoff(exp);
```

$$expandoff()$$

```
> expand(exp(a+b));
```

$$e^{(a+b)}$$

Vidíme, že vypnutím rozkladu pro exponenciální funkci se výše uvedený výraz nerozložil na součin dvou exponenciálních funkcí.

```
> expand(expandon()); expandon(exp);
```

$$expandon()$$

```
>  expand(exp(c+d));
```

$$e^c e^d$$

Naopak, když znovu nastavíme používání pravidel pro exponenciální funkci, tak se výše uvedený výraz rozloží na součin dvou exponenciálních funkcí.

## 2.3 Slučování výrazů

Příkaz `combine` se používá v případech, kdy chceme výraz o více členech obsažených v součtech, součinech a mocninách převést na výraz o jediném členu. Tento příkaz lze také aplikovat na seznamy a množiny výrazů. Může být volán s jedním nebo více parametry, kde první parametr je opět povinný a obsahuje upravovaný výraz. Další parametry mohou upřesňovat způsob, jakým má být transformace provedena. Těmito parametry mohou být například `trig`, `exp`, `ln`, `power`, `abs` nebo `icombine`. Pomocí těchto parametrů specifikujeme typy matematických funkcí, na něž má být příkaz `combine` aplikován. Použití a význam parametrů je obdobný jako v případě `simplify`.

**■ Příklad 2.4:**

V tomto příkladu uvedeme obvyklé způsoby použití funkcí `combine`:

```
> expand(sin(a+b),trig);
```

$$\sin(b)\cos(a)+\cos(b)\sin(a)$$

```
> combine(%,trig)=%;
```

$$\sin(b+a)=\sin(b)\cos(a)+\cos(b)\sin(a)$$

```
> 4*sin(x)^3=combine(4*sin(x)^3,trig);
```

$$4\sin(x)^3=-\sin(3x)+3\sin(x)$$

```
> exp(x)^2*exp(y)=combine(exp(x)^2*exp(y),exp);
```

$$\left(e^{x}\right)^2 e^{y}=e^{(2x+y)}$$

```
> combine(Int(x,x=a..b)-Int(x^2,x=a..b));
```

$$\int_a^b x-x^2\,dx$$

```
> combine(exp(sin(a)*cos(b))*exp(cos(a)*sin(b)),[trig,exp]);
```

$$e^{\sin(b+a)}$$

```
> combine(4^a * 6^b * 12^c * 5^d, 'power' );
```

$$4^a 6^b 12^c 5^d$$

```
> combine(4^a * 6^b * 12^c * 5^d, 'icombine' );
```

$$2^{(2a+b+2c)}3^{(c+b)}5^d$$

Příkaz `combine` s druhým parametrem `icombine` se pokouší sloučit součin mocnin celých čísel tak, že celá čísla rozloží na prvočinitele a jejich mocniny, zatímco `combine` s druhým parametrem `power` provede sloučení mocnin.

## 2.4 Rozklady polynomů

Rozklady polynomů na jejich kořenové činitele zajišťuje příkaz `factor`. Rozkládá polynomy o více neznámých s celočíselnými, reálnými nebo komplexními koeficienty na součin ireducibilních polynomů. Příkaz se volá s jedním nebo dvěma parametry. Jako první (povinný) parametr se uvádí výraz (polynom), který se má upravit, druhým parametrem je možné specifikovat číselnou množinu, nad kterou chceme výpočet provést. Pokud je jako první parametr příkazu `factor` zadána množina nebo seznam výrazů, jsou postupně rozkládány všechny prvky této množiny.

**■ Příklad 2.5:**

V tomto příkladu uvedeme obvyklé způsoby použití funkcí `factor` a `ifactor`:

```
> factor(6*x^2+18*x-24);
```

$$6\,(x+4)\,(x-1)$$

```
> factor(x^5-y^5);
```

$$(x-y)\,(x^4+x^3y+x^2y^2+xy^3+y^4)$$

```
> a^4-2=factor(a^4-2,sqrt(2));
```

$$a^4-2=(a^2-\sqrt{2})\,(a^2+\sqrt{2})$$

```
> seznam_vyrazu:=[x^2-16,x^4-16];
```

$$seznam\_vyrazu := [x^2-16,\ x^4-16]$$

```
> factor(seznam_vyrazu);
```

$$[(x-4)\,(x+4),\ (x-2)\,(x+2)\,(x^2+4)]$$

V případě, že nejde polynom rozložit v oboru celých čísel, je možno jej rozložit v oboru reálných nebo komplexních čísel.

```
> factor(x^3+5);
```

$$x^3+5$$

```
> factor(x^3+5.);
```

$$(x+1.709975947)\,(x^2-1.709975947\,x+2.924017740)$$

```
> factor(x^3+5,complex);
```

$$(x+1.709975947)\,(x+(-0.8549879733+1.480882610\,I))\,(x+(-0.8549879733-1.480882610\,I))$$

S rozklady polynomů na ireducibilní členy souvisí také problematika rozkladů celých čísel na součin prvočísel. To se provádí pomocí příkazu `ifactor`, neboť příkaz `factor` rozklad na součin prvočísel neprovede.

```
> factor(420);
```

$$420$$

```
> ifactor(420);
```

$$(2)^2\,(3)\,(5)\,(7)$$

## 2.5 Úpravy racionálních výrazů

Nejprve se zabývejme zjednodušováním racionálních výrazů obsahujících součty, součiny a celočíselné mocniny celých čísel a proměnných, které je realizováno funkcí `normal`. Tyto výrazy jsou převáděny na takzvanou *„normalizovanou formu“*, což je výraz v podobě zlomku, jehož čitatel i jmenovatel jsou pokud možno polynomy.

Příkaz `normal` může být volán s jedním nebo dvěma parametry. První parametr je povinný a obsahuje výraz na který má být funkce použita. Jako druhý parametr může být použito klíčové slovo `expanded`, kterým určujeme, zda polynomiální výrazy obsažené ve výsledku budou roznásobeny.

**■ Příklad 2.6:**

V tomto příkladu ukážeme použití příkazu `normal` jak s parametrem `expanded`, tak i bez něj:

```
> normal(x^2-(x+1)*(x-1)-1);
```

$$0$$

```
> (x^2-y^2)/(x-y)^3 = normal((x^2-y^2)/(x-y)^3);
```

$$\frac{x^2-y^2}{(x-y)^3}=\frac{x+y}{(x-y)^2}$$

```
> v:=1/x+x/(x+1);
```

$$v := \frac{1}{x}+\frac{x}{x+1}$$

```
> normal(v);
```

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

```
> normal(v,expanded);
```

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

Pomocí příkazu `normal` ovšem nelze upravovat výrazy obsahující zlomky s odmocninami ve jmenovateli. Pro tyto účely musíme použít příkaz `rationalize`, který převádí tyto zlomky na tvar, který již odmocniny ve jmenovateli nemá. Vyskytuje-li se odmocnina ve jmenovateli zlomku jako parametr jiné funkce (např. sinus), pak příkaz `rationalize` tuto odmocninu neodstraní.

**■ Příklad 2.7:**

Použití příkazu `rationalize` ukážeme na dvou příkladech.

```
> 2/(2-sqrt(2))=rationalize(2/(2-sqrt(2)));
```

$$\frac{2}{2-\sqrt{2}} = 2+\sqrt{2}$$

```
> (1+2^(1/3))/(1-2^(1/3));
```

$$\frac{1+2^{(1/3)}}{1-2^{(1/3)}}$$

```
> rationalize(%);
```

$$-\left(1+2^{(1/3)}\right)\left(1+2^{(1/3)}+2^{(2/3)}\right)$$

## 2.6 Konverze mezi typy výrazů

Potřebujeme-li některý výsledek použít pro další výpočet v jiném tvaru, nabízí se široké možnosti příkazu `convert` pro převody výrazů do požadovaného tvaru. Některé konverze provedené tímto příkazem jsou pouze převody mezi datovými typy. Jedná se například o převody mezi číselnými soustavami nebo ze stupňů na radiány. Jinou skupinu konverzí pomocí příkazu `convert` tvoří převody algebraických výrazů.

Tento příkaz se obvykle volá se dvěma parametry. První parametr obsahuje výraz, který má být upraven, druhý parametr specifikuje formu, na kterou má být výraz převeden.

**■ Příklad 2.8:**

Pro snazší pochopení příkazu `convert` jsou v tomto příkladu uvedeny ukázky použití tohoto příkazu pro převod čísla do binárního tvaru, součet prvků v seznamu, převod výrazu obsahujícího desetinné číslo na výraz obsahující zlomek a rozklad zlomku na parciální zlomky.

```
> convert(9,binary);
```

$$1001$$

```
> convert([1,2,3,4],`+`);
```

$$10$$

```
> convert(1.23456*x,fraction);
```

$$\frac{3858}{3125}x$$

```
> (x^3+x)/(x^2-1)=convert((x^3+x)/(x^2-1),parfrac,x);
```

$$\frac{x^3+x}{x^2-1} = x+\frac{1}{x-1}+\frac{1}{x+1}$$

## 2.7 Další základní operace

V této kapitole si probereme ještě několik základních příkazů, jejichž znalost může značně zjednodušit práci se systémem Maple.

### 2.7.1 Příkaz `isolate`

Častým případem při manipulaci s matematickými výrazy bývá situace, kdy potřebujeme z rozsáhlého výrazu nebo rovnice vyjádřit některý v nich obsažený podvýraz (podvýrazem se samozřejmě rozumí i jednotlivé proměnné). v takovém případě je vhodné použít příkaz `isolate`, který daný výraz nebo rovnici převede do tvaru, kde je na levé straně daný podvýraz a na pravé straně jeho hodnota vyjádřená pomocí ostatních proměnných z výrazu nebo rovnice.

Příkaz `isolate` má tři parametry, z nichž první dva jsou povinné. První parametr obsahuje výraz respektive rovnici, z níž se bude podvýraz vyjadřovat. Pokud je v tomto parametru uveden výraz, předpokládá se implicitně, že má být roven nule. Jako druhý parametr musí být uveden podvýraz, který má být z rovnice vyjádřen. Třetím nepovinným parametrem může být přirozené číslo označující maximální počet úprav, které mohou být při výpočtu provedeny.

**■ Příklad 2.9:**

Zde je ukázáno použití příkazu `isolate`.

```
> isolate((x-b),x);
```

$$x = b$$

```
> isolate(4*x*sin(x)=3,sin(x));
```

$$\sin(x) = \frac{3}{4\,x}$$

```
> isolate(x^2-3*x-5,x^2);
```

$$x^2 = 3\,x + 5$$

### 2.7.2 Funkce `assign` a `unassign`

Názvy těchto funkcí určují jejich význam, ale je vhodné si objasnit jejich přesný smysl. Použití:

```
> assign (proměnné, výraz);
```

má stejný význam jako

```
> proměnná := výraz;
```

Ovšem s tím rozdílem, že první parametr procedury `assign` je vyhodnocen, přičemž při použití operátoru `:=` k tomuto jevu nedochází. Co je na `assign` nejzajímavější je to, že může být rekurzivně aplikováno na libovolnou množinu rovností, jaká vychází například jako výsledek použití funkce `solve`. Pokud chceme v takové situaci přiřadit řešení do odpovídajících proměnných, pak je nejvhodnější použití funkce `solve`

**■ Příklad 2.10:**

Zde si na názorném příkladu osvětlíme použití příkazu `assign`. Výsledkem příkazu pro řešení rovnic `solve` (viz kapitola 3) je řešení `{y=6, x=5}`. Hodnoty hledaných proměnných `x` a `y` nejsou inicializovány, proto použijeme příkaz `assign`, který tyto hodnoty dosadí. Obdobně lze tento příkaz použít u funkce `dsolve`.

```
> Results := solve ( {x+y=a, b*x-y=c}, {x,y} );
```

$$Results := \left\{ ?\ y = \frac{b\,a - c}{b+1}, x = \frac{c+a}{b+1}\ ? \right\}$$

```
> x,y;
```

$$x, y$$

```
> assign (Results);
> x,y;
```

$$\frac{c+a}{b+1}, \frac{b\,a - c}{b+1}$$

Funkce `unassign` slouží pro zrušení obsahu proměnných a inicializaci na prázdnou hodnotu. Této možnosti využíváme hlavně v těch případech, kdy nechceme následující výpočet zaplnit jmény proměnných, které jsme naplnili a již nebudeme potřebovat. Použití této funkce a další způsoby nové inicializace proměnných si projdeme v další kapitole.

### 2.7.3 Nevyhodnocený výraz

Výrazy, které jsou uzavřeny v *apostrofech* `''` nejsou vyhodnocovány. Pokud tedy zadáme `a := 1; b := 4; f := 'a + b'`, potom výsledná hodnota `f` bude `a+b` a nikoliv hodnota 5.

**■ Příklad 2.11:**

Použití zpožděného přiřazení ukážeme na několika příkladech:

```
> a := 1; b := 4; f := 'a + b';
```

$$a := 1$$

$$b := 4$$

$$f := a + b$$

Při každém vyhodnocení je potom jedna vnější dvojice *apostrofů* `''` odstraněna:

```
> 'f'; f;
```

$$f$$

$$5$$

```
> ''a''; 'a'; a;
```

$$'a'$$

$$a$$

$$1$$

Specifickým případem nevyhodnoceného výrazu je zrušení obsahu proměnné a její inicializace do základního stavu, podobně jako ve funkci `unassign`.

**■ Příklad 2.12:**

V tomto příkladu si ukážeme dva způsoby zrušení obsahu proměnné. Při použití funkce `unassign` je důležité nezapomenout použít pro danou proměnnou nevyhodnoceného výrazu, neboť jinak Maple provede vyhodnocení a vloží obsah proměnné místo jejího jména.

```
> x := 5; x := 'x'; x;
```

$$x := 5$$

$$x := x$$

$$x$$

```
> x := 5; unassign ('x'); x;
```

$$x := 5$$

$$x$$

```
> x := 5; unassign (x);
```

$$x := 5$$

```
Error, (in unassign) cannot unassign `5' (argument must be as-
signable)
```

# Kapitola 3

# Řešení rovnic a nerovnic

Stejně jako při „ručním“ řešení rovnic a nerovnic pomocí tužky a papíru existují i při jejich řešení s využitím systémů počítačové algebry dva základní přístupy. Buď můžeme hodnotu neznámé vypočítat (a to přesně či numericky) nebo se můžeme uchýlit ke grafickému řešení rovnice. Ve druhém případě je výsledkem grafické znázornění hodnot řešení vyhovujících dané rovnici či soustavě rovnic. V praxi se nejčastěji tyto hodnoty řešení zobrazují v systému kartézských souřadnic. Následující text je věnován řešení rovnic a nerovnic pomocí Maple. Rovnice budeme řešit jak analyticky, tak numericky (v případě, že nelze analytické řešení vyjádřit ) a rovněž pro větší názornost i graficky.

## 3.1 Analytické řešení rovnic a nerovnic

Podobně jako v ostatních oblastech matematiky, nabízí Maple 9.5 i v případě řešení rovnic několik příkazů, které můžeme použít. Ne všechny příkazy jsou však vhodné pro všechny typy rovnic.

### 3.1.1 Příkaz isolate

Tomuto příkazu byla již věnována část předcházející kapitoly, kde se příkaz `isolate` používal pro vytknutí matematických výrazů obsažených ve výrazech. Omezíme-li se při obecném chápání příkazu `isolate` pouze na vyjadřování jednotlivých proměnných, získáme tak nástroj pro řešení rovnic.

**■ Příklad 3.1:**

Nejprve ukážeme řešení lineární rovnice, kde je řešením racionální číslo, které Maple 9.5 vyjádří přesně pomocí zlomku:

```
> isolate(3*x-7=0,x);
```

$$x = \frac{7}{3}$$

V případě lineárních rovnic se symbolickými koeficienty můžeme pomocí funkce `isolate` získat i jejich obecné řešení:

```
> isolate(a*x+b=0,x);
```

$$x = -\frac{b}{a}$$

Použití příkazu `isolate` pro řešení kvadratických rovnic, případně dalších algebraických rovnic vyšších řádů, již není zcela úspěšné, neboť získáme jen jedno řešení. Příkaz `isolate` určí pouze jeden kořen zadané rovnice a další kořen již nehledá. Stejně se tento příkaz chová i v ostatních případech, kdy rovnice mají více kořenů.

```
> isolate(a*x^2+b*x+c=0,x);
```

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

```
> isolate(x^2+3*x+2=0,x);
```

$$x = -1$$

■ **Příklad 3.2:**

Příkaz `isolate` lze rovněž s úspěchem použít při řešení dalších typů rovnic, jako jsou například některé goniometrické nebo exponenciální rovnice.

```
> isolate(sin(2*x)=1,x);
```

$$x = \frac{1}{4}\pi$$

```
> isolate(exp(2*x)=4,x);
```

$$x = \ln(2)$$

### 3.1.2 Funkce solve

Nejvhodnějším nástrojem pro řešení rovnic v systému Maple 9.5 je příkaz `solve`. Hlavní jeho výhodou vůči příkazu `isolate` je možnost řešit soustavy rovnic o více neznámých. Příkaz `solve` také na rozdíl od `isolate` hledá i další kořeny u těch rovnic, které mají více řešení. Jeho další výhodou je schopnost řešit nerovnice, což `isolate` neumožňuje.

K příkazu `solve` existuje několik jeho dalších variant (`msolve`, `isolve`, `fsolve`, `dsolve`, `rsolve`), které jsou použitelné v různých specifických případech. V této kapitole popíšeme základní variantu příkazu `solve`, příkaz `fsolve` bude popsán v kapitole o numerickém řešení rovnic. Podrobný popis ostatních zmiňovaných příkazů je uveden v nápovědě systému Maple 9.5.

Příkaz `solve` se obvykle volá se dvěma parametry. Prvním parametrem je (ne)rovnice nebo soustava (ne)rovnic, která se má řešit, druhým je proměnná nebo proměnné, které chceme z (ne)rovnic vypočítat. Zadáme-li jako první parametr matematický výraz nebo proceduru, implicitně se položí roven nule a takto vzniklá rovnice se dále řeší. Pokud nezadáme druhý parametr, rovnice se automaticky vyřeší pro všechny proměnné, které obsahuje. V případě, kdy rovnice obsahuje více než jednu proměnnou, získáme někdy pro některou z proměnných nic neříkající výsledek typu $x=x$.

■ **Příklad 3.3:**

Na příkladu hledání řešení pro proměnnou $b$ jsou ukázány rozdílné výsledky, které získáme bez použití, případně s použitím druhého parametru funkce `solve`. Při řešení rovnice s číselnými koeficienty Maple 9.5 dává přesně řešení, kde se nám vyskytuje ve zlomku i odmocnina. Chceme-li získat přibližné řešení použijeme k vyhodnocení řešení funkce `evalf`.

```
> solve(a+b=c);
```

$$\{c = c,\ b = b,\ a = -b + c\}$$

```
> solve(a+b=c,b);
```

$$-a + c$$

```
> solve(6*x-3=sqrt(2)*x);
```

$$-\frac{3}{-6+\sqrt{2}}$$

```
> evalf(%);
```

$$0.6541953144$$

V případě řešení soustavy (ne)rovnic zadává se jako první parametr příkazu `solve` množina (ne)rovnic nebo výrazů a jako druhý parametr množina proměnných. Množina v Maple 9.5 s zapisuje jako posloupnost jejich prvků do složených závorek. Druhý parametr příkazu `solve` můžeme případně opět vynechat. Výsledek příkazu je pak vrácen jako množina rovnic, kde její prvky mají tvar *proměnná* = *výraz*.

Pro přístup k těmto řešením pak použijeme příkaz **`assign`**. Jak bylo zmíněno, tento příkaz zpracovává množinu dvojic *proměnná* = *výraz* a pro každou takovou dvojici provede přiřazení *proměnná* := *výraz*. Pomocí této funkce můžeme vložit výsledky řešení soustavy rovnic do jednotlivých proměnných odpovídajících hledaným neznámým. Ukážeme si to na následujícím příkladě:

**■ Příklad 3.4:**

Vyřešme soustavu tří lineárních rovnic:

```
> soustava_rovnic:={u+v+w=1, 3*u+v=3, u-2*v-w=0};
```

$$soustava\_rovnic := \{u + v + w = 1, 3\,u + v = 3, u - 2\,v - w = 0\}$$

```
> solve(soustava_rovnic);
```

$$\left\{ v = \frac{3}{5}, w = \frac{-2}{5}, u = \frac{4}{5} \right\}$$

```
> assign (%);
> u; v; w;
```

$$\frac{4}{5}$$

$$\frac{3}{5}$$

$$\frac{-2}{5}$$

Jak již bylo naznačeno výše, můžeme příkaz `solve` úspěšně použít také pro obecné vyjádření kořenů algebraických rovnic.

**■ Příklad 3.5:**

Ukážeme na tomto příkladu, že na rozdíl od příkazu `isolate` nalezne příkaz `solve` všechny kořeny obecné kvadratické a kubické rovnice.

```
> solve(a*x^2+b*x+c,x);
```

$$\frac{-b + \sqrt{b^2 - 4\,a\,c}}{2\,a}, \frac{-b - \sqrt{b^2 - 4\,a\,c}}{2\,a}$$

```
> solve(x^3+p*x^2+q*x+r,{x});
```

$$\left\{ x = \frac{1}{6}\left( 36\,q\,p - 108\,r - 8\,p^3 + 12\sqrt{12\,q^3 - 3\,q^2\,p^2 - 54\,q\,p\,r + 81\,r^2 + 12\,r\,p^3}\right)^{(1/3)} \right.$$

$$- \frac{6\left(\frac{1}{3}q - \frac{1}{9}p^2\right)}{\left( 36\,q\,p - 108\,r - 8\,p^3 + 12\sqrt{12\,q^3 - 3\,q^2\,p^2 - 54\,q\,p\,r + 81\,r^2 + 12\,r\,p^3}\right)^{(1/3)}}$$

$$\left. - \frac{1}{3}p \right\},$$

$$\left\{ x = \right.$$

$$- \frac{1}{12}\left( 36\,q\,p - 108\,r - 8\,p^3 + 12\sqrt{12\,q^3 - 3\,q^2\,p^2 - 54\,q\,p\,r + 81\,r^2 + 12\,r\,p^3}\right)^{(1/3)}$$

$$+ \frac{3\left(\frac{1}{3}q - \frac{1}{9}p^2\right)}{\left( 36\,q\,p - 108\,r - 8\,p^3 + 12\sqrt{12\,q^3 - 3\,q^2\,p^2 - 54\,q\,p\,r + 81\,r^2 + 12\,r\,p^3}\right)^{(1/3)}}$$

$$- \frac{1}{3}p$$

$$+ \frac{1}{2}I\sqrt{3}\left( \frac{1}{6}\left( 36\,q\,p - 108\,r - 8\,p^3 + 12\sqrt{12\,q^3 - 3\,q^2\,p^2 - 54\,q\,p\,r + 81\,r^2 + 12\,r\,p^3}\right)^{(1/3)} \right.$$

$$\left. + \frac{6\left(\frac{1}{3}q - \frac{1}{9}p^2\right)}{\left( 36\,q\,p - 108\,r - 8\,p^3 + 12\sqrt{12\,q^3 - 3\,q^2\,p^2 - 54\,q\,p\,r + 81\,r^2 + 12\,r\,p^3}\right)^{(1/3)}} \right)$$

$$\left\{ \right\},\quad \left\{ \right.$$

$$x = -\frac{1}{12}\left(36qp - 108r - 8p^3 + 12\sqrt{12q^3 - 3q^2p^2 - 54qpr + 81r^2 + 12rp^3}\right)^{(1/3)}$$

$$+ \frac{3\left(\frac{1}{3}q - \frac{1}{9}p^2\right)}{\left(36qp - 108r - 8p^3 + 12\sqrt{12q^3 - 3q^2p^2 - 54qpr + 81r^2 + 12rp^3}\right)^{(1/3)}} - \frac{1}{3}p$$

$$- \frac{1}{2}I\sqrt{3}\left(\frac{1}{6}\left(36qp - 108r - 8p^3 + 12\sqrt{12q^3 - 3q^2p^2 - 54qpr + 81r^2 + 12rp^3}\right)^{(1/3)}\right.$$

$$\left. + \frac{6\left(\frac{1}{3}q - \frac{1}{9}p^2\right)}{\left(36qp - 108r - 8p^3 + 12\sqrt{12q^3 - 3q^2p^2 - 54qpr + 81r^2 + 12rp^3}\right)^{(1/3)}}\right)$$

$$\left. \right\}$$

**■ Příklad 3.6:**

Na příkladu rovnice 4. řádu ukážeme jak lze ukládat její řešení do proměnné `reseni` typu seznam, který se v Maple zapisuje do hranatých závorek. Pak lze jednotlivá řešení používat jako prvky seznamu, např. když chceme zkontrolovat správnost třetího řešení jeho dosazením do původní rovnice.

```
> rovnice := x^4-5*x^2+6*x=2;
```

$$rovnice := x^4 - 5x^2 + 6x = 2$$

```
> reseni:=[solve(rovnice,x)];
```

$$reseni := [1, 1, \sqrt{3} - 1, -1 - \sqrt{3}]$$

```
> evalf(reseni);
```

$$[1., 1., 0.732050808, -2.732050808]$$

```
> subs( x=reseni[3], rovnice );
```

$$(\sqrt{3} - 1)^4 - 5(\sqrt{3} - 1)^2 + 6\sqrt{3} - 6 = 2$$

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

$$2 = 2$$

K tomu jsme využili mapleovské funkce `subs`, která do svého druhého parametru - výrazu `rovnice` - dosadí za proměnnou `x` třetí řešení rovnice 4. řádu uložené v prvku seznamu `reseni[3]`.

**■ Příklad 3.7:**

Pomocí příkazu `solve` lze řešit i trigonometrické rovnice:

```
> solve(cos(x)^2=1, x);
```

$$0, \pi$$

```
> solve( sin(x)=cos(x)-1, x );
```

$$-\frac{1}{2}\pi, 0$$

```
> solve( cos(x)+y = 9, x );
```

$$\pi - \arccos(y - 9)$$

**■ Příklad 3.8:**

V tomto příkladu ukážeme způsob použití příkazu `solve` při řešení soustav lineárních a kvadratických nerovnic:

```
> solve({ x+y > 0, x-y <= 1, y = 2 }, {x,y});
```

$$\{y = 2, -2 < x, x \le 3\}$$

```
> solve( {x^2-x-2<=0,x^2-x>0}, x );
```

$$\{-1 \le x, x < 0\}, \{1 < x, x \le 2\}$$

Příkazy typu solve hledají řešení rovnic v celém komplexním oboru. Chceme-li se omezit na řešení pouze v oboru reálném, použijeme k tomu knihovnu `RealDomain` jedním z následujících způsobů:

```
> solve(x^3+x=0,x);
```

$$0, I, -I$$

```
> use RealDomain in solve(x^3+x=0,x);
> end;
```

$$0$$

```
> ln(-1);
```

$$I\pi$$

```
> RealDomain[ln](-1);
```

$$undefined$$

```
> with(RealDomain);
Warning, these protected names have been redefined and unpro-
tected: Im, Re, ^, arccos, arccosh, arccot, arccoth, arccsc,
arccsch, arcsec, arcsech, arcsin, arcsinh, arctan, arctanh,
cos, cosh, cot, coth, csc, csch, eval, exp, expand, limit, ln,
log, sec, sech, signum, simplify, sin, sinh, solve, sqrt,
surd, tan, tanh
```

[ℑ, ℜ, ^, arccos, arccosh, arccot, arccoth, arccsc, arccsch, arcsec, arcsech, arcsin, arcsinh, arctan, arctanh, cos, cosh, cot, coth, csc, csch, *eval*, exp, *expand*, *limit*, ln, log, sec, sech, signum, *simplify*, sin, sinh, *solve*, sqrt, *surd*, tan, tanh]

```
> ln(-1);
```

$$undefined$$

Jak bylo již zmíněno, pomocí příkazu `solve` lze řešit i nerovnice. Výsledek příkazu je pak realizován ve formě intervalů, které jsou uvozeny klíčovým slovem `RealRange`. Otevřené hranice intervalů jsou pak vyznačeny klíčovým slovem `Open`:

```
> solve(x^4-5*x^2+4<=0);
```

$$RealRange(-2, -1), RealRange(1, 2)$$

```
> solve(x^4-5*x^2+4<0);
```

$$RealRange(Open(-2), Open(-1)), RealRange(Open(1), Open(2))$$

## 3.2 Numerické řešení rovnic

V případě, že Maple není schopen najít přesné řešení rovnice či soustavy rovnic, pak lze tuto úlohu řešit numericky nebo graficky. Základním nástrojem je příkaz `fsolve`. Přesnost výsledku je pak dána nastavením systémové proměnné `Digits`. Větší přesnost řešení však znamená prodloužení doby výpočtu. Příkaz `fsolve` řeší úlohu některou z iteračních numerických metod (podle zvolené metody) a tudíž nalezne pouze jeden kořen rovnice (s výjimkou funkcí, u kterých lze některé kořeny nalézt přesně). Tento nedostatek je však vyvážen mnoha volbami parametrů příkazů `fsolve`, kdy můžeme specifikovat interval hledání, počáteční aproximaci či dokonce kořeny, kterým se chceme vyhnout.

Volby parametrů příkazu `fsolve` jsou demonstrovány na následujících příkladech, kde rovnice nelze vyřešit analyticky a kde nám příkaz `solve` dá jako výsledek řešení například výraz ve funkci `RootOf`:

```
> solve(23*x^5 + 105*x^4 - 10*x^2 + 17*x=0,x);
```

$$0, RootOf(23\_Z^4 + 105\_Z^3 - 10\_Z + 17, index = 1),$$
$$RootOf(23\_Z^4 + 105\_Z^3 - 10\_Z + 17, index = 2),$$

$$\mathrm{RootOf}(23\_Z^4 + 105\_Z^3 - 10\_Z + 17, index = 3),\ \mathrm{RootOf}(23\_Z^4 + 105\_Z^3 - 10\_Z + 17, index = 4)$$

```
> fsolve(23*x^5 + 105*x^4 - 10*x^2 + 17*x=0,x,-1..1);
```

$$-.6371813185, 0.$$

```
> fsolve(23*x^5 + 105*x^4 - 10*x^2 + 17*x=0,x,-1..1, avoid=
{x=0});
```

$$-.6371813185$$

```
> solve({sin(x+y) - exp(x)*y = 0,x^2 - y = 2},{x,y});
```

$$\left\{ x = \mathrm{RootOf}\left( \sin(\_Z + \_Z^2 - 2) - \mathbf{e}^{\_Z}\,\_Z^2 + 2\,\mathbf{e}^{\_Z} \right),\ y = \mathrm{RootOf}\left( \sin(\_Z + \_Z^2 - 2) - \mathbf{e}^{\_Z}\,\_Z^2 + 2\,\mathbf{e}^{\_Z} \right)^2 - 2 \right\},$$

$$\left\{ x = -1 - \mathrm{RootOf}\left( \sin(\_Z + \_Z^2 - 2) - \mathbf{e}^{\_Z}\,\_Z^2 + 2\,\mathbf{e}^{\_Z} \right),\ y = 2\,\mathrm{RootOf}\left( \sin(\_Z + \_Z^2 - 2) - \mathbf{e}^{\_Z}\,\_Z^2 + 2\,\mathbf{e}^{\_Z} \right) + \mathrm{RootOf}\left( \sin(\_Z + \_Z^2 - 2) - \mathbf{e}^{\_Z}\,\_Z^2 + 2\,\mathbf{e}^{\_Z} \right)^2 - 1 \right\}$$

```
> fsolve({sin(x+y) - exp(x)*y = 0,x^2 - y = 2},{x,y});
```

$$\{x = -6.017327250, y = 34.20822723\}$$

```
> fsolve({sin(x+y) - exp(x)*y = 0,x^2 - y = 2},{x,y},{x=-
1..1,y=-2..0});
```

$$\{y = -1.552838698, x = -.6687012050\}$$

## 3.3 Grafické řešení rovnic a nerovnic

Maple má rozsáhlé možnosti pro zobrazování grafů funkcí. Toho lze úspěšně využít při grafickém řešení rovnic a nerovnic. Zde nám Maple 9.5 nabízí možnost, jak rychle a přesně zobrazit daný objekt, což lze využít i v případě nelineárních rovnic, K tomu využijeme příkazy standardní grafické knihovny `plots`.

Velkou výhodou tohoto přístupu je také možnost zobrazovat grafy i bez předchozího hlubšího studia ostatních příkazů systému Maple 9.5. Většina jeho zobrazovacích možností je rychle a snadno dostupná. Nejjednodušším způsobem jak vytvořit graf funkce, je kliknout pravým tlačítkem myši na vybranou funkci a z kontextového menu Maple 9.5 vybrat položku `plots`. Program již sám automaticky nastaví ostatní parametry grafu (souřadnice, barvy, rozsahy …), které však lze kdykoliv změnit podle potřeby.

### 3.2.1 Grafické řešení rovnic

Grafické řešení rovnic spočívá v tom, že do jednoho grafu umístíme jak křivku dané funkce, tak i body, které vyhovují řešení její rovnice. v případě rovnic o jedné neznámé, jejichž řešení zobrazujeme V dvourozměrných grafech v kartézských souřadnicích, umísťujeme kořeny rovnice zpravidla na osu *x*. Má-li například nějaká rovnice kořen 1, bude tento kořen znázorněn jako bod o souřadnicích [1,0]. V následujícím příkladu je ukázán postup při vizualizaci řešení kvadratické rovnice.

**■ Příklad 3.9:**

Pokud chceme v Maple 9.5 vykreslovat grafy, je nutné nejdříve vyvolat grafickou knihovnu `plots` pomocí následujícího příkazu:

```
> with(plots):
Warning, the name changecoords has been redefined

> f:=x^2+2*x-3;
```

$$f := x^2 + 2x - 3$$

```
> k := [solve(f)];
```

$$k := [1, -3]$$

Předchozí dva příkazy uložily do proměnné `f` rovnici (výraz) a nalezly její kořeny. Kořeny rovnice se uložily jako seznam do proměnné `k`. Jak je vidět, rovnice má dva kořeny o hodnotách 1 a -3. Do proměnné `pf` uložíme graf dané funkce a do proměnné `pk` pak uložíme grafické znázornění kořenů rovnice.

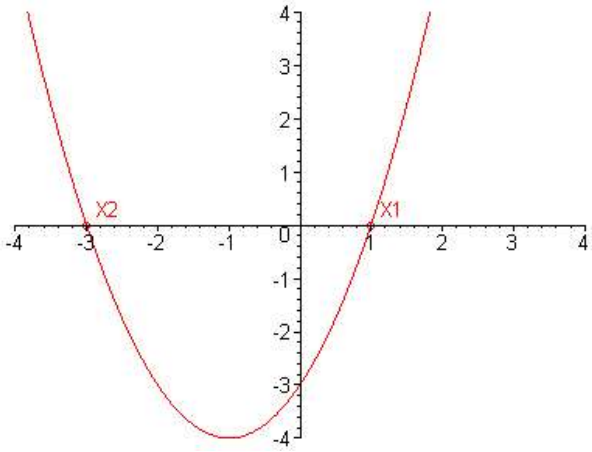
```
> pf:=plot(f,x=-4..4,y=-4..4):
> pk:=plot([[k[1],0],[k[2],0]],style=point,symbol=circle):
```

Do proměnných `pt1` a `pt2` uložíme popisky kořenů rovnice, které se mají zobrazit. v tomto případě jsou kořeny označeny jako X1 a X2. Popisky kořenů jsou zde posunuty ve směru obou os o hodnotu 0,3. To je z důvodu, aby se popisky nepřekrývaly s křivkou a osami souřadnic.

```
> pt1:=textplot([k[1]+0.3,0.3,'X1']):
> pt2:=textplot([k[2]+0.3,0.3,'X2']):
```

Na závěr se vše zobrazí do jednoho grafu pomocí příkazu `display`.

```
> display({pf,pk,pt1,pt2});
```



■ **Příklad 3.10:**

Na tomto příkladu ukážeme postup při řešení rovnice *tg (x + 1) – 1 = 0* na intervalu <-1,1>.

Nejprve připomeňme, že funkce *tg* není na celé reálné ose spojitá a má body nespojitosti (dokonce 2. druhu) a v Maple má název `tan`. Proto pro její korektní zobrazení je nutné u mapleovské funkce `plot` nastavit parametr `discont`, kterým identifikujeme nespojitost zobrazované funkce na hodnotu `true`.

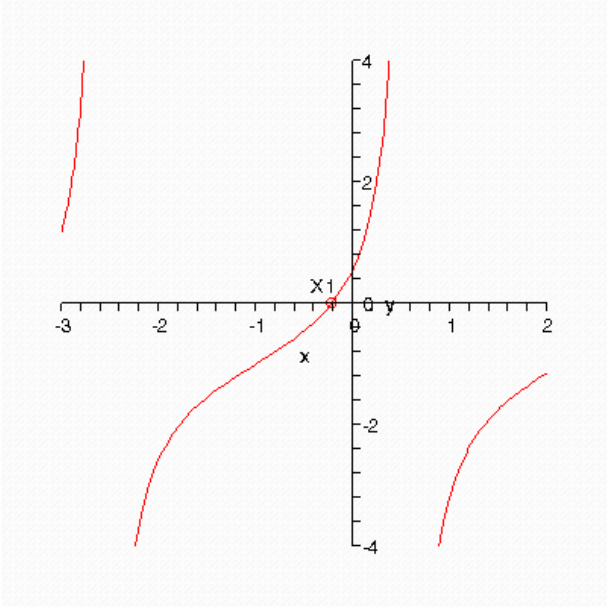
```
> with(plots):
Warning, the name changecoords has been redefined

> f:=tan(x+1)-1;
```

$$f := \tan(x+1) - 1$$

```
> k:=[solve(f)];
```

$$k := \left[\frac{1}{4}\pi - 1\right]$$

```
> pf:=plot(f,x=-3..2,y=-4..4, discont=true):
> pk:=plot([[k[1],0]],style=point,symbol=circle):
> pt:=textplot([k[1]-0.1,0.3,'X1']):
> display({pf,pk,pt});
```




### 3.2.1 Grafické řešení nerovností – příkaz `inequal`

V Maple9.5 umožňuje zobrazit graficky řešení soustavy lineárních nerovností dvou proměnných v rovině pomocí příkazu `inequal`. Počítačový graf řešení této soustavy sestává ze čtyř částí, kde u každá části je možno si zvolit její barvu pomocí speciálního parametru. Tyto parametry jsou:

- `optionsfeasible` – pro oblast řešení, která splňuje všechny nerovnosti,
- `optionsexcluded` – pro oblasti, kde není splněna alespoň jedna nerovnost,
- `optionsclosed` - pro přímky, reprezentující hranice ostrých nerovností,

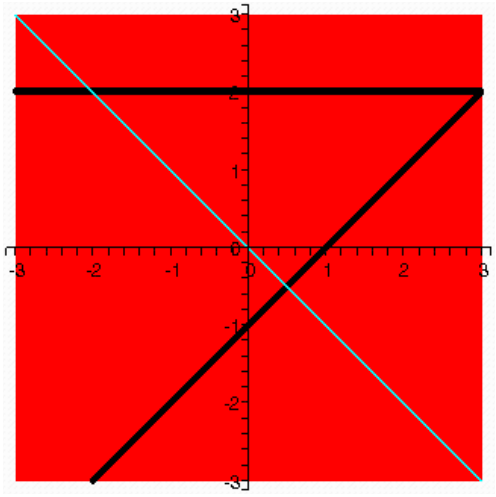
- `optionsopen` - pro přímky, reprezentující hranice neostrých nerovností a průsečíky těchto přímek.

Styly zobrazení těchto čtyř oblastí se mohou volit nezávisle na sobě. Ukážeme to v následujícím příkladu.

**■ Příklad 3.11:**

Nejprve nalezneme grafické řešení soustavy nerovností z příkladu 3.8

```
> with(plots): inequal( { x+y > 0, x-y <= 1, y = 2 }, x=-3..3,
y=-3..3,optionsfeasible=(color=red), optionsopen=(color=blue,
thickness=2), optionsclosed=(color=green, thickness=3),
optionsexcluded=(color=yellow) );
```

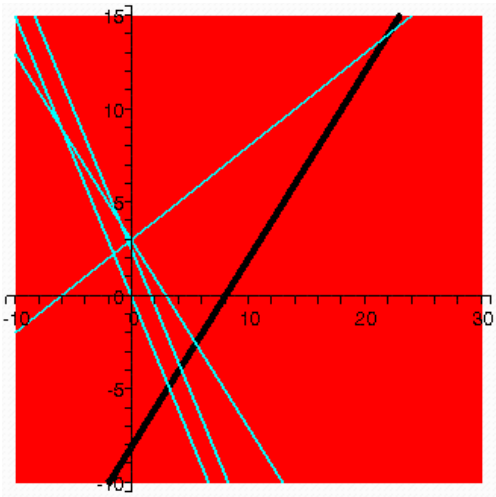


Jako další příklad vyřešíme následující soustavu nerovností

```
> soustava_nerovnosti:={a+b>3, 2*b-a<6, 3*a+2*b>5, -b+a<=8,
3*a+2*b>0};
```

$$soustava\_nerovnosti := \{2b - a < 6, -b + a \le 8, 5 < 3a + 2b, 3 < a + b, 0 < 3a + 2b\}$$

```
> inequal( soustava_nerovnosti,a=-10..30, b=-10..15, opti-
onsfeasible=(color=red), optionsopen=(color=blue,
thickness=2), optionsclosed=(color=green, thickness=3),
optionsexcluded=(color=yellow));
```



# Kapitola 4

## Matematická analýza v systému Maple

V oblasti matematické analýzy nabízí Maple pestrou škálu příkazů. Z této nabídky se omezíme na výpočty limit, derivací a integrálů, řešení obyčejných a parciálních diferenciálních rovnic.. Dále se omezíme na funkce jedné proměnné. Složitější části matematické analýzy budou stručně zmíněny na konci kapitoly. Podrobnější výklad a užití Maplu v matematické analýze lze nalézt v (Plch, Došlá, Sojka 1999), (Došlá, Plch, Sojka 2002).

## 4.1 Výpočet limit

Pro výpočet limity funkce slouží v systému Maple příkaz `limit`. V základní podobě má příkaz `limit` dva parametry. První parametr udává funkci nebo výraz, pro něž se limita počítá. Druhý parametr definuje hodnotu nezávisle proměnné, pro níž se limita počítá. Při výpočtu nevlastních limit se užívá maplovská konstanta `infinity` a kladné respektive záporné ∞ se označuje jako `+infinity` resp. `-infinity`.

**■ Příklad 4.1:**

V tomto příkladu se ukazuje užití příkazu `limit` se dvěma základními parametry.

```
> limit(sin(x)/x, x=0);
```

$$1$$

```
> limit(exp(x), x=infinity);
```

$$\infty$$

```
> limit(exp(x), x=-infinity);
```

$$0$$

```
> limit(1/x, x=0);
```

$$\mathit{undefined}$$

Jak je vidět, v případě lomené funkce z předcházejícího příkladu limita v bodě $x=0$ neexistuje. Pro tuto funkci však existují v bodě $x=0$ limita zprava i zleva. Výpočet limity zprava nebo zleva se v příkazu `limit` uvádí pomocí třetího nepovinného parametru. Pomocí tohoto parametru můžeme také vybrat číselný obor (reálná nebo komplexní čísla),ve kterém se má limita počítat.

**■ Příklad 4.2:**

Použití třetího parametru si opět ukážeme na lomené funkci.

```
> limit(1/x, x=0, left);
```

$$-\infty$$

```
> limit(1/x, x=0, right);
```

$$\infty$$

```
> limit(1/x, x=0, real);
```

$$\mathit{undefined}$$

```
> limit(1/x, x=0, complex);
```

$$\infty - \infty \,]$$

Pro zefektivnění výstupů můžeme kombinovat příkaz `limit` s příkazem `Limit`, který danou limitu vypíše v „tradiční“ matematické formě. Parametry příkazu `Limit` jsou totožné s parametry `limit`.

**■ Příklad 4.3:**

Příklad ukazuje použití příkazu `Limit`.

```
> Limit(2/(3*x), x=infinity)=limit(2/(3*x), x=infinity);
```

$$\lim_{x \to \infty} \left( \frac{2}{3x} \right) = 0$$

S funkcí `Limit` lze pracovat jako s matematickým výrazem. Je možné ji například upravovat pomocí příkazu `combine`.

```
> combine(Limit(1/x,x=0)*Limit(x,x=0));
```

$$\lim_{x \to 0} 1$$

## 4.2 Výpočet derivace

Chceme-li v Maple derivovat nějakou funkci nebo výraz, je to možno provést pomocí příkazu `diff` nebo diferenciálního operátoru `D`. U příkazu **`diff`** se jako první parametr uvádí funkce nebo výraz, který chceme derivovat. Jako další parametry se uvádí nezávisle proměnné, podle nichž se bude derivovat. V případě derivací funkce jedné proměnné to znamená, že kolikrát danou proměnnou uvedeme, tolikátou derivaci bude Maple počítat. Chceme-li například spočítat třetí derivaci funkce *f*, bude mít příkaz `diff` tvar `diff(f,x,x,x)`. Případně můžeme použít zkrácenou formu zápisu `diff(f,x$3)` s využitím operátoru `$`, jehož popis najdeme v nápovědě.

**■ Příklad 4.4:**

Příklad ukazuje výpočet 1., 2. a 3. derivace funkce *sin(x)*.

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

$$\cos(x)$$

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

$$-\sin(x)$$

```
> diff(sin(x),x$3);
```

$$-\cos(x)$$

Stejně jako k příkazu `limit` existuje příkaz `Limit`, existuje i k příkazu `diff` příkaz `Diff`. Parametry obou příkazů jsou opět shodné.

**■ Příklad 4.5:**

V tomto příkladě ukážeme, jak je vhodné používat příkaz `Diff` na derivování výrazu podle dvou nezávisle proměnných `x` a `a`:

```
> Diff((a*x)^7-sin(a*x),x$2)=diff((a*x)^7-sin(a*x),x$2);
```

$$\frac{\partial^2}{\partial x^2}(a^7 x^7 - \sin(a\,x)) = 42\,a^7 x^5 + \sin(a\,x)\,a^2$$

```
> Diff((a*x)^7-sin(a*x),a$2)=diff((a*x)^7-sin(a*x),a$2);
```

$$\frac{\partial^2}{\partial a^2}(a^7 x^7 - \sin(a\,x)) = 42\,a^5 x^7 + \sin(a\,x)\,x^2$$

Pro derivování funkcí jedné proměnné lze použít diferenciální operátor `D`. Jeho syntaxe je jednodušší (podrobnosti naleznete v nápovědě pomocí příkazu `?D`) a umožňuje kratší zápis příkazu pro derivování funkcí. Argumentem je pouze název derivované funkce:

```
> D(sin);
```

$$\cos$$

```
> D(ln);
```

$$x \to \frac{1}{x}$$

Výsledkem aplikace operátoru D je opět funkce jedné proměnné, tj. `D(f)(x) = diff(f(x), x)`. Tzn., že `D(f) = unapply(diff(f(x), x), x)`, kde maplovský příkaz `unapply` slouží k definování funkcí a definuje první parametr (algebraický výraz) jako funkci druhého parametru. Takže operátor D zobrazuje funkce jedné proměnné opět na funkce jedné proměnné.

Použití operátoru `D` demonstrujeme na následujících příkladech:

```
> D(sin)(0);
```

$$1$$

```
> D(ln+sin)(x);
```

$$\frac{1}{x} + \cos(x)$$

```
> D(ln*sin)(x);
```

$$\frac{\sin(x)}{x} + \ln(x)\cos(x)$$

Složení dvou funkcí *f(x)* a *g(x)*, tj. *(f.g)(x) = f(g(x))*, se v Maplu zapisuje pomocí operátoru @, tj. `f@g` označuje složení funkcí `f` a `g`.

Použití operátoru `D` na složení funkcí demonstrujeme na následujících příkladech:

```
> D(ln@sin);
```

$$x \to \frac{1}{x} @ \sin \cos$$

Operátor @ zde značí skládání funkcí, a to ve vstupu i ve výstupu.

```
> D(ln@sin)(x);
```

$$\frac{\cos(x)}{\sin(x)}$$

```
> (D@@3)(x->x^4);
```

$$x \to 24\,x$$

## 4.3 Výpočet integrálů

Výpočet integrálů v systému Maple se provádí pomocí příkazu `int`. Tento příkaz se použije jak v případě, kdy chceme spočítat neurčitý integrál, tak i v případě, kdy se jedná o výpočet určitého integrálu. V prvním případě je výsledkem příkazu primitivní funkce, ve druhém případě Maple dává konkrétní hodnotu nebo výraz (obsahuje-li integrovaná funkce parametry).

Nejdříve si ukážeme způsob, jakým se pomocí příkazu `int` počítá neurčitý integrál. Příkaz `int` se volá se dvěma parametry. Prvním parametrem je integrovaná funkce nebo výraz, druhým parametrem je proměnná, podle které se integruje. Maple vypisuje výsledky bez aditivní konstanty.

**■ Příklad 4.6:**

Výpočet neurčitého integrálu:

```
> int(-x^2+3,x);
```

$$-\frac{1}{3}x^3+3\,x$$

```
> int(2+exp(x),x);
```

$$2\,x+e^x$$

Pokud chceme spočítat pomocí příkazu `int` určitý integrál, přiřadíme k proměnné ve druhém parametru integrační interval. Význam prvního parametru zůstává stejný jako v případě neurčitého integrálu.

**■ Příklad 4.7:**

Výpočet určitého integrálu:

```
> int(-x^2+3,x=-1..1);
```

$$\frac{16}{3}$$

```
> int(2+exp(x),x=1..2);
```

$$-e+e^2+2$$

Také k příkazu `int` existuje příkaz `Int` a má stejný význam jako u ostatních příkazů `Limit` a `Diff`.

**■ Příklad 4.8:**

Použití příkazu `Int`:

```
> Int(sin(x),x)=int(sin(x),x);
```

$$\int \sin(x)\,dx = -\cos(x)$$

```
> Int(-x^2+3,x=-1..1)=int(-x^2+3,x=-1..1);
```

$$\int_{-1}^{1} -x^2+3\,dx = \frac{16}{3}$$

```
> Int(2+exp(x),x=1..2)=int(2+exp(x),x=1..2);
```

$$\int_{1}^{2} 2+e^{x}\,dx = -e+e^{2}+2$$

## 4.4 Obyčejné diferenciální rovnice

Diferenciální rovnice a jejich systémy jsou jedním z nejdůležitějších prostředků k řešení praktických úloh při modelování přírodovědných (biologických, fyzikálních) a technických jevů. Maple poskytuje účinné nástroje pro řešení všech typů diferenciálních rovnic. Základním nástrojem je příkaz `dsolve`, který má velké množství volitelných argumentů. V této kapitole se omezíme na jeho základní použití pro řešení jednoduchých diferenciálních rovnic v obecném tvaru i s počátečními podmínkami (Cauchyova počátečního problému). Povinným parametrem příkazu `dsolve` je diferenciální rovnice (je-li uveden jen výraz, pokládá se roven nule). Pro vyjádření derivace se užívá příkaz `diff`. Počáteční podmínky jsou společně s rovnicí uzavřeny do množinových závorek. Výsledkem příkazu `dsolve` je analytické řešení (pokud existuje) v uzavřeném tvaru, kde jsou uvedeny konstanty _C1, _C2, atd. V případě, že jsou předepsány počáteční podmínky, tak řešení je uvedeno v uzavřeném tvaru, kde příslušné konstanty jsou již vyčísleny z počátečních podmínek. Ukážeme to na několika příkladech.

**■ Příklad 4.9:**

Řešení nehomogenní lineární rovnice prvního řadu obecně a s předepsanou počáteční podmínkou:

```
> linrov := sin(x)*diff(y(x),x)-cos(x)*y(x)=cos(x);
```

$$linrov := \sin(x)\left(\frac{d}{dx}y(x)\right) - \cos(x)\,y(x) = \cos(x)$$

```
> dsolve({linrov});
```

$$\{y(x) = -1 + \_C1\sin(x)\}$$

```
> dsolve({linrov,y(Pi/2)=0});
```

$$y(x) = -1 + \sin(x)$$

**■ Příklad 4.10:**

Řešení homogenní rovnice druhého řadu obecně a s předepsanými počátečními podmínkami:

```
> difrov:=diff(y(x),x,x)+diff(y(x),x)-y(x)/x=0;
```

$$difrov := \left(\frac{d^2}{dx^2}y(x)\right) + \left(\frac{d}{dx}y(x)\right) - \frac{y(x)}{x} = 0$$

```
> dsolve(difrov);
```

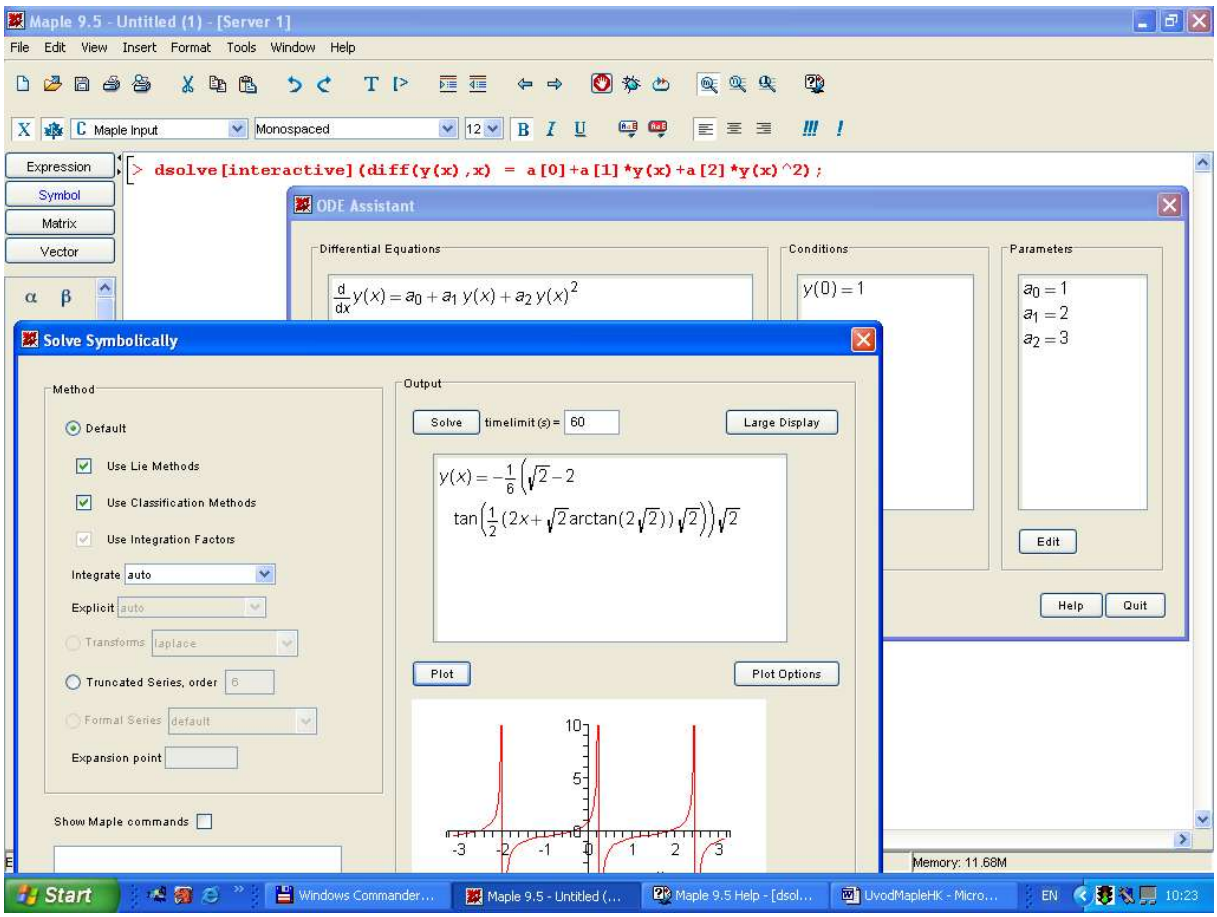
$$y(x) = \left( \left( -\frac{e^{(-x)}}{x} + \mathrm{Ei}(1, x) \right) \_C1 + \_C2 \right) x$$

```
> dsolve({difrov,y(0)=0,D(y)(0)=1});
```

$$y(x) = x$$

Podrobný rozbor volitelných argumentů a jejich význam najdeme v nápovědě k příkazu `dsolve`.K příkazu dsolve existuje jeho interaktivní varianta `dsolve[interactive]`, která využívá nový grafický interface Maplu, který je vytvořen pomocí Mapletů, které jsou popsány v dalších kapitolách. V tomto příkazu lze předepsat jaké výsledky, týkající se řešení diferenciální rovnice chceme zobrazit. Ukážeme to na příkladu řešení diferenciální rovnice:

$$y'(x)=a_0+a_1 y(x)+a_2 y(x)^2$$




*Obrázek č. 4.1: Interaktivní řešení diferenciální rovnice $y'(x)=a_0+a_1 y(x)+a_2 y(x)^2$*

Z obr. 4.1 je vidět, že když jsme zvolili ve výše uvedené rovnici konstanty $a_0=1$, $a_1=2$, $a_2=3$ a počáteční podmínku $y(0)=1$, tak jsme dostali analytické řešení této rovnice v uzavřeném tvaru a jeho průběh jsme si zobrazili na zvoleném intervalu $\langle -\pi,\pi \rangle$.

Pokud by nám široké možnosti těchto příkazů nestačily, je možno použít pro řešení diferenciálních rovnic balík `DEtools`. Asi nejužitečnější z tohoto balíku je příkaz `odeadvisor`, který dokáže k dané rovnici najít její typ a tím napovědět metodu jejího symbolického nebo numerického řešení. Ukážeme to opět na několika příkladech:

```
> with(DEtools):
```

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

$$\sin(x)\left(\frac{d}{dx}y(x)\right) - \cos(x)\,y(x) = 0$$

```
> odeadvisor(%);
```

$$[\_separable]$$

```
> diff(y(x),x)*diff(y(x),x,x,x,x)-diff(y(x),x,x)*diff(y(x),
x,x,x) + diff(y(x),x)^3*diff(y(x),x,x,x);
```

$$\left(\frac{d}{dx}y(x)\right)\left(\frac{d^4}{dx^4}y(x)\right) - \left(\frac{d^2}{dx^2}y(x)\right)\left(\frac{d^3}{dx^3}y(x)\right) + \left(\frac{d}{dx}y(x)\right)^3\left(\frac{d^3}{dx^3}y(x)\right)$$

```
> odeadvisor(%);
```

$$[[\_high\_order, \_missing\_x], [\_high\_order, \_missing\_y],$$
$$[\_high\_order, \_with\_linear\_symmetries]]$$

Tuto možnost lze využít při procvičování řešení diferenciálních rovnic, ale zejména tím lze urychlit budoucí výpočet s využitím příkazu `dsolve` – typ rovnice je jedním z jeho volitelných parametrů, který říká, která pravidla se pro řešení mají použít a Maple pak nemusí uvažovat všechny možné postupy řešení diferenciálních rovnic. Pro podrobný výčet funkcí balíku `DEtools` použijte nápovědu např. `?DEtools`.

Dalším důležitým příkazem týkajícím se řešení obyčejných diferenciálních rovnic je příkaz `odetest` – ověření řešení. Jeho prvním argumentem je řešení a druhým rovnice. Tento příkaz vrací hodnotu 0 pro správné řešení a v případě špatného řešení vrací nenulovou funkci, která vznikne jako rozdíl levé a pravé strany po dosazení.

```
> rov:=diff(diff(y(x),x),x)*diff(y(x),x)*y(x)*x^6-
2*diff(y(x),x)^3*x^6+2*diff(y(x),x)^2*y(x)*x^5+y(x)^5;
```

$$rov := \left(\frac{d^2}{dx^2}y(x)\right)\left(\frac{d}{dx}y(x)\right)y(x)\,x^6 - 2\left(\frac{d}{dx}y(x)\right)^3 x^6 + 2\left(\frac{d}{dx}y(x)\right)^2 y(x)\,x^5 + y(x)^5$$

```
> sol:=dsolve(rov);
```

$$sol := y(x) = 0,\ y(x) = \frac{3\,x^3}{(2\,x - 2\,\_C1\,x^2)^{(3/2)} + 3\,\_C2\,x^3},$$

$$y(x) = \frac{3\,x^3}{-(2\,x - 2\,\_C1\,x^2)^{(3/2)} + 3\,\_C2\,x^3}$$

```
> odetest(sol[1],rov);
```

$$0$$

```
> odetest(sol[2],rov);
```

$$0$$

```
> odetest(sol[3],rov);
```

$$0$$

```
> odetest(y(x)=x,rov);
```

$$x^5$$

## 4.5 Parciální diferenciální rovnice

Ještě častějším případem při řešení praktických úloh z oblasti přírodovědných i technických věd jsou diferenciální rovnice s parciálními derivacemi funkcí více proměnných podle jejich jednotlivých proměnných. Těmto rovnicím se říká parciální diferenciální rovnice a Maple zde nabízí podobné možnosti jako pro obyčejné diferenciální rovnice:

Základním nástrojem pro řešení parciálních diferenciálních rovnic je příkaz `pdsolve`. Má podobnou syntaxi jako `dsolve` popsaný v předchozí části, liší se jak v použití příkazu `diff` při sestavování rovnic, kdy předepisujeme parciální derivaci dle příslušných nezávisle proměnných, tak v mnohem komplikovanějším zadávání okrajových a počátečních podmínek. Jeho podrobný popis získáme opět pomocí nápovědy (`?pdsolve`), kde je uvedena i strategie použití tohoto příkazu pro nalezení obecného řešení parciální diferenciální rovnice, případně nalezení quasiobecného řešení, atd.

```
> pdsolve(diff(f(x,y),y)*diff(arctan(x^(1/2)*y),y)
+ diff(f(x,y),x)*diff(arctan(x^(1/2)*y),x)=0);
```

$$f(x,y) = \_F1(-2x^2+y^2)$$

Podobně jako v předchozí kapitole rozšiřuje možnosti příkazu `pdsolve` balík `PDEtools`. Jeho použití je podobné jako u balíku `DEtools` a lze jej nastudovat z nápovědy. Funkce mají dokonce podobné názvy – řešení ověříme příkazem `pdetest`. Novinkou je příkaz `difforder` zjišťující jak celkový řád rovnice, tak i řád jednotlivých proměnných:

```
> with(PDEtools):
> rov := diff(f(x,y,z), x$2, z$3) = diff(f(x,y,z), y$5, x);
```

$$rov := \frac{\partial^5}{\partial x^2\, \partial z^3} f(x,y,z) = \frac{\partial^6}{\partial x\, \partial y^5} f(x,y,z)$$

```
> difforder(rov);
```

$$6$$

```
> difforder(rov,x),difforder(rov,y),difforder(rov,z);
```

$$2, 5, 3$$

## 4.6 Analýza v komplexním oboru

Příkazy Maplu na pro výpočet limit, derivací či integrálů komplexních funkcí komplexní proměnné jsou stejné jako pro funkce v reálném oboru a akceptují „komplexní“ argumenty.

Maple však obsahuje i několik příkazů, jejichž výsledky jsou teoreticky odvoditelné pouze v komplexním oboru – například rezidua funkcí, singularity funkcí a zobecnění jejich Taylorových rozvojů na tzv. Laurentovy rozvoje.

Pro Taylorovy rozvoje v případě reálných funkcí, ale i pro Laurentovy rozvoje komplexních funkcí se používá příkaz `series`. Jeho prvním argumentem je funkce, druhým pak proměnná a bod rozvoje zapsaný ve formě rovnice a nepovinným třetím parametrem je počet členů (Maple má standardně nastaven rozvoj se třemi členy).

```
> series(x^3/(x^4+4*x-5),x=infinity);
```

$$\frac{1}{x} - \frac{4}{x^4} + \frac{5}{x^5} + O\left(\frac{1}{x^7}\right)$$

Koeficient u *1/x* v Laurentově rozvoji se nazývá reziduum funkce a jedná se o pojem důležitý pro mnoho výpočtů. Jeho hodnotu získáme v Maplu pomocí příkazu `residue`, jehož prvním argumentem je opět funkce a ve druhém argumentu zadáme proměnnou a bod výpočtu.

```
> residue(1/(x^4+6*x^3+13*x^2+12*x+4),x=-1);
```

$$-2$$

Z těchto rozvojů lze snadno odvodit singularity funkce, a to obecně v komplexním případě. Pro výpočet singularit použijeme příkaz `singular`, jehož jediným argumentem je daná funkce. Jedná-li se o periodickou singularitu, použije Maple pro vyjádření novou proměnnou uvozenou podtržítkem (znak za podtržítkem odpovídá značení číselného oboru, do kterého proměnná patří).

```
> singular(1/sin(I*x));
```

$$\{x = I\pi \_ZI\sim\}$$

## 4.7 Minimalizace a optimalizace funkcí

Nástroje matematické analýzy se užívají také v oblasti výpočtů extrémů funkcí jedné nebo více proměnných. Maple má nástroje k výpočtu extrémů jak volných, tak i vázaných rovnicemi či nerovnicemi a může nám výrazně pomoci při řešení například i ekonomických úloh.

Pro nalezení minima dané funkce jedné či více proměnných použijeme příkaz `minimize` a pro výpočet maxima příkaz `maximize`. Chceme-li vypočítat volné globální minimum (maximum) funkce, zadáme pouze název funkce. V případě vázáného extrému, kde jsou omezení zadány ve tvaru rovnic, tak příslušné rovnice zadáme jako další parametr v příkazu. Můžeme také specifikovat oblast, ve které extrém hledáme, ve formě intervalů. Pomocí parametru `location` se vyžaduje kromě optimální hodnoty funkce i bod, ve kterém extrém nastal. Tyto možnosti jsou ilustrovány následujícím příkladem:

```
> maximize(-x^2+3*x-y^2-3*y-3);
```

$$\frac{3}{2}$$

```
> minimize(x^2-3*x+y^2+3*y+3, x=2..4, y=-4..-2, location);
```

$$-1, \{[\{x = 2, y = -2\}, -1]\}$$

Druhou možností pro výpočet extrémů je příkaz `extrema`, který nerozlišuje mezi minimy a maximy. Pokud má výraz minimální i maximální hodnotu, tak pomocí tohoto příkazu nalezneme jejich hodnoty. Výhody tohoto přístupu spočívají například v obecných výpočtech pro funkce s parametry, kdy pro některou hodnotu parametru nabývá funkce maximum a pro některou minimum – při použití příkazu `minimize` nebo `maximize` bychom museli specifikovat interval, ve kterém parametr hledáme. Nevýhodou tohoto příkazu je fixní počet argumentů, který nás nutí zadat prázdnou množinu omezujících podmínek v případě, že hledáme volný extrém. Stejně jako u předchozích příkazů i zde lze zadat omezující podmínky ve tvaru rovnic:

```
> extrema( a*x^2+b*x+c,{},x );
```

$$\left\{ \frac{-b^2 + 4\,c\,a}{4\,a} \right\}$$

*Poznámka:* Příkaz `extrema` nalezl hodnotu extrému kvadratické funkce a ne hodnotu proměnné `x` v níž nastává extrém.

```
> extrema(x^2+y^2-z,x^2+y^2+z^2=1,{x,y,z});
```

$$\left\{ -1, \frac{5}{4} \right\}$$

V tomto příkladu byly nalezeny minimální a maximální hodnota výrazu na jednotkové kouli v trojrozměrném euklidovském prostoru.

Dosud jsme se zde nezabývali případem, kdy je extrém vázán omezujícími podmínkami ve tvaru nerovnic. Zde Maple poskytuje řešení problému pro lineární funkce více proměnných a pro omezení ve tvaru lineárních nerovnic. Toto řešení není založeno na prostředcích matematické analýzy a je dostupné v balíku `simplex`. Pozor – funkce se jmenují stejně jako ve standardní knihovně! V tomto případě dostaneme namísto optimální hodnoty bod, ve kterém nastává extrém.

```
> with(simplex):
Warning, the protected names maximize and minimize have been
redefined and unprotected

> maximize(x+y,{4*x+3*y<=5,3*x+4*y<=4});
```

$$\left\{ y = \frac{1}{7}, x = \frac{8}{7} \right\}$$

```
> minimize(4*x+5*y+3*z,{x+y+z>=12,2*x+3*y+z>=25,x+10*y+z>=30},
NONNEGATIVE);
```

$$\{y = 2, x = 9, z = 1\}$$

# Kapitola 5

## Lineární algebra v systému Maple

V této kapitole se seznámíme se základními možnostmi Maplu pro práci s vektory a maticemi. Naučíme se používat základní nástroje pro řešení systémů lineárních rovnic v maticovém tvaru a pro počítání determinantů, vlastních čísel a podobných charakteristik matice.

### 5.1 Práce s vektory a maticemi v různých knihovnách Maple

K vektorům a maticím má Maple tři různé přístupy. Základním příkazem je k vytvoření matice nebo vektoru v Maplu příkaz `array`, pomocí kterého se definuje podobně jako v Pascalu datová struktura pole (podrobněji viz nápovědu `?array`). V jeho parametrech zadáváme nejprve rozměry pole a poté příslušné hodnoty prvků pole. Nevýhodou je, že hodnoty musí přesně odpovídat rozměrům pole, a proto například matici musíme naplnit dvakrát vnořeným seznamem hodnot.

Pro operace s maticemi a vektory se v Maple používá příkaz `evalm`, kterému zadáme k vyhodnocení maticový výraz s aritmetickými operacemi sčítání, odečítání, násobení a také inverze matice. Pozor – násobení matic je operací v mnoha ohledech (například i co se týče komutativity) rozdílnou od násobení čísel, proto se ve maticovém výrazu značí dvojznakem &* (klasické násobení pouze hvězdičkou)!

Použití příkazu `evalm` ilustruje následující příklad:

```
> A:=array(1..3,1..3,[[1,2,3],[4,5,6],[7,8,9]]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
> B:=array(1..3,1..3,[[2,4,-3],[1,0,3],[-3,1,-2]]);
```

$$B := \begin{bmatrix} 2 & 4 & -3 \\ 1 & 0 & 3 \\ -3 & 1 & -2 \end{bmatrix}$$

```
> v:=array(1..10,[seq(i,i=1..10)]);
```

$$v := [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$$

```
> evalm(A&*B+2*A-3*B);
```

$$\begin{bmatrix} -9 & -1 & 12 \\ 0 & 32 & -6 \\ 18 & 50 & 9 \end{bmatrix}$$

Možnosti Maplu pro operace s maticemi rozšiřují balíky `linalg` a `LinearAlgebra`. Oba nabízejí podobnou škálu příkazů pro oblast lineární algebry. V současné době je více doporučován pro operace s maticemi balík `LinearAlgebra`, který má mnohonásobně rychlejší operace než jsou v balíku `linalg`. Proto se na tento balík omezíme, podrobný popis operací v balíku `linalg` najdeme v nápovědě.

V balíku `LinearAlgebra` najdeme příkazy pro téměř všechny úlohy týkající se lineární algebry. S tím však souvisí o něco vyšší náročnost zápisu jeho maticových operací a také delší názvy příkazů. Matice a vektory zde zadáváme pomocí speciálních znaků < (menší než), > (větší než) a | (roura). Znaky „menší než“ a „větší než“ slouží jako uzavírací závorky – tj. pro vektor budou jednoduché, pro matici dvakrát vnořené. Znak „roura“ zde nahrazuje čárku pro oddělení jednotlivých sloupců (pozor – matice se zde zadává po sloupcích!).

Součet matic se provádí pomocí příkazu `MatrixAdd` (s variantou `VectorAdd` pro vektory) a násobení dvou matic pomocí příkazu `MatrixMatrixMultiply`. Existují i jeho varianty `MatrixVectorMultiply` pro násobení vektoru a matice, dále `MatrixScalarMultiply` pro násobení matice číslem (skalárem), `VectorScalarMultiply` pro násobení vektoru číslem (skalárem) a `DotProduct` pro skalární součin).

Všechny tyto příkazy lze provést pomocí jediného příkaz `Add` pro lineární kombinace matic, vektorů a skalárů. Tento příkaz se přizpůsobí podle toho, jestli jsou jeho parametrem matice, vektory či skaláry. Jeho podrobný rozbor lze nalézt v nápovědě. Rozdíly oproti předchozím variantám práce s maticemi demonstruje následující příklad:

```
> with(LinearAlgebra):
> A:=<<1,4,7>|<2,5,8>|<3,6,9>>;
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
> C:=<<2,4,-3>|<0,1,-1>|<-1,0,2>>;
```

$$C := \begin{bmatrix} 2 & 0 & -1 \\ 4 & 1 & 0 \\ -3 & -1 & 2 \end{bmatrix}$$

```
> MatrixAdd(A,C);
```

$$\begin{bmatrix} 3 & 2 & 2 \\ 8 & 6 & 6 \\ 4 & 7 & 11 \end{bmatrix}$$

```
> Add(A,C);
```

$$\begin{bmatrix} 3 & 2 & 2 \\ 8 & 6 & 6 \\ 4 & 7 & 11 \end{bmatrix}$$

```
> MatrixMatrixMultiply(A,C);
```

$$\begin{bmatrix} 1 & -1 & 5 \\ 10 & -1 & 8 \\ 19 & -1 & 11 \end{bmatrix}$$

Příkazy z balíku `LinearAlgebra` umožňují generovat zvolenou matici ze soustavy rovnic či automaticky naplnit jednotkovou nebo nulovou matici. Mezi základnější maticové příkazy z balíku `LinearAlgebra`, jejichž popis lze nalézt v nápovědě patří: `RowDimension`, `ColumnDimension`, `Dimension`, `Transpose`, `Rank`, `GaussianElimination`, `NullSpace`.

Jejich použití ukážeme v následujících příkladech:

```
> B:=<<2,3,-1>|<5,0,-2>|<-1,-4,0>|<2,1,5>>;
```

$$B := \begin{bmatrix} 2 & 5 & -1 & 2 \\ 3 & 0 & -4 & 1 \\ -1 & -2 & 0 & 5 \end{bmatrix}$$

```
> RowDimension(B);
```

$$3$$

```
> ColumnDimension(B);
```

$$4$$

```
> Dimension(B);
```

$$3, 4$$

```
> GaussianElimination(B);
```

$$\begin{bmatrix} 2 & 5 & -1 & 2 \\ 0 & \frac{-15}{2} & \frac{-5}{2} & -2 \\ 0 & 0 & \frac{-2}{3} & \frac{88}{15} \end{bmatrix}$$

```
> Transpose(B);
```

$$\begin{bmatrix} 2 & 3 & -1 \\ 5 & 0 & -2 \\ -1 & -4 & 0 \\ 2 & 1 & 5 \end{bmatrix}$$

```
> A:=<<1,4,7>|<2,5,8>|<3,6,9>>;
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
> Rank(A);
```

$$2$$

```
> NullSpace(A);
```

$$\left\{\begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}\right\}$$

Balík `LinearAlgebra` obsahuje další příkazy pro manipulaci s jednotlivými sloupci či řádky, které lze nalézt v nápovědě a nebudeme je zde uvádět.

Mezi velmi důležité příkazy pro získání důležitých charakteristik matice patří: `Determinant`, `^(-1)` (operátor pro výpočet inverze čtvercové matice), `Eigenvalues`, `Eigenvectors`, `JordanForm`.

Jejich použití ukážeme v následujících příkladech:

```
> C:=<<2,4,-3>|<0,1,-1>|<-1,0,2>>;
```

$$C := \begin{bmatrix} 2 & 0 & -1 \\ 4 & 1 & 0 \\ -3 & -1 & 2 \end{bmatrix}$$

```
> Determinant(C);
```

$$5$$

```
> C^(-1);
```

$$\begin{bmatrix} \frac{2}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{-8}{5} & \frac{1}{5} & \frac{-4}{5} \\ \frac{-1}{5} & \frac{2}{5} & \frac{2}{5} \end{bmatrix}$$

```
> M:=<<1,2,0>|<0,0,3>|<1,1,1>>;
```

$$M := \begin{bmatrix} 1 & 0 & 1 \\ 2 & 0 & 1 \\ 0 & 3 & 1 \end{bmatrix}$$

```
> Eigenvalues(M);
```

$$\begin{bmatrix} 3 \\ -\frac{1}{2}+\frac{1}{2}I\sqrt{3} \\ -\frac{1}{2}-\frac{1}{2}I\sqrt{3} \end{bmatrix}$$

```
> Eigenvectors(M);
```

$$\left[\begin{array}{c} -\frac{1}{2}+\frac{1}{2}I\sqrt{3} \\ -\frac{1}{2}-\frac{1}{2}I\sqrt{3} \\ 3 \end{array}\right], \left[\begin{array}{ccc} \frac{1}{-\frac{3}{2}+\frac{1}{2}I\sqrt{3}} & \frac{1}{-\frac{3}{2}-\frac{1}{2}I\sqrt{3}} & \frac{1}{2} \\ -\frac{\frac{1}{3}I\sqrt{3}}{-\frac{1}{2}+\frac{1}{2}I\sqrt{3}} & \frac{\frac{1}{3}I\sqrt{3}}{-\frac{1}{2}-\frac{1}{2}I\sqrt{3}} & \frac{2}{3} \\ 1 & 1 & 1 \end{array}\right]$$

```
> JordanForm(M);
```

$$\left[\begin{array}{ccc} 3 & 0 & 0 \\ 0 & -\frac{1}{2}+\frac{1}{2}I\sqrt{3} & 0 \\ 0 & 0 & -\frac{1}{2}-\frac{1}{2}I\sqrt{3} \end{array}\right]$$

Na závěr této kapitoly uvedeme příklady s dalšími důležité příkazy: `GramSchmidt`, `Basis`, pro práci s vektory:

```
> v1:=<1,2,0>;v2:=<1,1,1>;v3:=<-1,0,2>;
```

$$v1 := \left[\begin{array}{c} 1 \\ 2 \\ 0 \end{array}\right]$$

$$v2 := \left[\begin{array}{c} 1 \\ 1 \\ 1 \end{array}\right]$$

$$v3 := \left[\begin{array}{c} -1 \\ 0 \\ 2 \end{array}\right]$$

```
> GramSchmidt({v1,v2,v3});
```

$$\left\{\left[\begin{array}{c} 1 \\ 2 \\ 0 \end{array}\right], \left[\begin{array}{c} -\frac{4}{5} \\ \frac{2}{5} \\ 2 \end{array}\right], \left[\begin{array}{c} \frac{2}{3} \\ -\frac{1}{3} \\ \frac{1}{3} \end{array}\right]\right\}$$

```
> GramSchmidt({v1,v2,v3},normalized);
```

$$\left\{ \begin{bmatrix} \frac{1}{5}\sqrt{5} \\ \frac{2}{5}\sqrt{5} \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{1}{3}\sqrt{6} \\ -\frac{1}{6}\sqrt{6} \\ \frac{1}{6}\sqrt{6} \end{bmatrix}, \begin{bmatrix} -\frac{1}{15}\sqrt{30} \\ \frac{1}{30}\sqrt{30} \\ \frac{1}{6}\sqrt{30} \end{bmatrix} \right\}$$

```
> v1:=<1,2,3>;v2:=<4,5,6>;v3:=<7,8,9>;
```

$$v1 := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$v2 := \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

$$v3 := \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix}$$

```
> Basis({v1,v2,v3});
```

$$\left\{ \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \right\}$$

Kromě toho můžeme v balíku `LinearAlgebra` počítat i báze průniku a součtu vektorových prostorů nebo odchylku dvou vektorů. Pro podrobnější popis těchto možností, stejně jako další širokou škálu funkcí tohoto balíku doporučujeme použít nápovědu prodavších více než sto dalších příkazů.

## 5.2 Řešení systémů lineárních rovnic v různých knihovnách Maple

V této kapitole se budeme věnovat řešení systémů lineárních rovnic. Stejně jako v předchozí části popíšeme standardní možnosti Maple i obou balíků `linalg` a `LinearAlgebra`.

Pro řešení systémů lineárních rovnic je možno použít příkaz `solve`, uvedený v třetí kapitole. V případě systémů rovnic mu zadáme dva argumenty – prvním je množina rovnic a druhým je množina neznámých. V případě, že zadaný systém lineárních rovnic nemá řešení, tak příkaz `solve` má prázdný výstup, v případě nekonečně mnoha řešení `solve` nechá některé neznámé jako parametry a řešení vyjádří s jejich pomocí . Ilustrujme tyto možnosti na příkladech:

```
> solve({3*x+y+z=10,x-y+2*z=1,x+y-3*z=2},{x,y,z});
```

$$\{x = 2, z = 1, y = 3\}$$

```
> solve({3*x+y+z=10,x-y+2*z=1,x+y-3*z=2,x+y+z=0},{x,y,z});
> solve({3*x+y+z=10,x-y+2*z=1},{x,y,z});
```

$$\left\{ z = z, y = \frac{7}{4} + \frac{5}{4} z, x = \frac{11}{4} - \frac{3}{4} z \right\}$$

V balíku `LinearAlgebra` je základním příkazem pro řešení systémů lineárních rovnic příkaz `LinearSolve`. Jeho prvním (a jediným povinným) parametrem však musí být rozšířená matice soustavy lineárních rovnic včetně pravých stran. Lze ji zadat buď ručně nebo pohodlněji pomocí generátoru maticového zápisu ze soustavy rovnic (viz nápověda).

Volbou volitelných parametrů pak můžeme ušetřit čas i paměť při výpočtu řešení tím, že specifikujeme podrobněji úlohu – například uvedeme řídkost matice nebo doporučíme Maplu vhodnost použití některého rozkladu (Choleského, QR rozkladu, LU rozkladu apod.) matice při řešení. Opět to ukážeme na příkladech:

```
> with(LinearAlgebra):
> M:=<<3,1,1>|<1,-1,1>|<1,2,-3>|<10,1,2>>;
```

$$M := \begin{bmatrix} 3 & 1 & 1 & 10 \\ 1 & -1 & 2 & 1 \\ 1 & 1 & -3 & 2 \end{bmatrix}$$

```
> LinearSolve(M);
```

$$\begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

```
> M:=<<3,1,1,1>|<1,-1,1,1>|<1,2,-3,1>|<10,1,2,0>>;
```

$$M := \begin{bmatrix} 3 & 1 & 1 & 10 \\ 1 & -1 & 2 & 1 \\ 1 & 1 & -3 & 2 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

```
> LinearSolve(M);
Error, (in LinearSolve) inconsistent system

> M:=<<3,1>|<1,-1>|<1,2>|<10,1>>;
```

$$M := \begin{bmatrix} 3 & 1 & 1 & 10 \\ 1 & -1 & 2 & 1 \end{bmatrix}$$

```
> LinearSolve(M);
```

$$\begin{bmatrix} \frac{11}{4} - \frac{3}{4}\_t0_3 \\ \frac{7}{4} + \frac{5}{4}\_t0_3 \\ \_t0_3 \end{bmatrix}$$

Z výše uvedeného příkladu vidíme, že v případě, že systém lineárních rovnic nemá řešení tak namísto prázdného výstupu, který nám dá příkaz `solve`, příkaz `LinearSolve` nás na to upozorní chybovým hlášením.

Balík `LinearAlgebra` obsahuje kromě tohoto základního příkazu pro řešení soustavy rovnic také mnoho dalších příkazů upravujících matici na horní či dolní trojúhelníkový tvar a širokou škálu rozkladů matice. Podrobněji o této problematice se lze dočíst v nápovědě.

# Kapitola 6

## Algebraické výpočty v systému Maple

Tato kapitola nás seznámí s možnostmi Maplu pro řešení problémů z oblastí kombinatoriky, teorie grup a čísel.

### 6.1 Kombinatorika

Často se setkáváme s úlohami, kdy je nutné prozkoumat všechny možné kombinace některých jevů nebo pracovat s pořadím prvků, čili permutacemi. Pro tyto účely je v Maplu příkaz `factorial`, který počítá faktoriál daného nezáporného čísla nebo celočíselného výrazu anebo zobecněný faktoriál pro čísla nebo výrazy typu `float` pomocí funkce `GAMMA`. Tato příkaz lze ve zkráceném zápisu nahradit vykřičníkem. Pro počítání binomických koeficientů je příkaz `binomial` (má dva argumenty). Vzhledem k tomu, že Maple není v podstatě omezen délkou čísla (omezení naráží až na délku výpisu či množství paměti), tak je schopen vypočítat faktoriál teoreticky libovolně velkého čísla..

```
> factorial(10);
```

$$3628800$$

```
> 20!;
```

$$2432902008176640000$$

```
> factorial(-2.1);
```

$$9.7148063\,83$$

```
> factorial(5.2);
```

$$169.4060995$$

```
> binomial(49,6);
```

$$13983816$$

```
> binomial(2, 1/2);
```

$$\frac{16}{3\,\pi}$$

```
> binomial(2.1, 2+3*I);
```

$$-56.56167619 - 98.27156511\,I$$

```
> binomial(n, 2);
```

$$\mathrm{binomial}(n, 2)$$

```
> expand(%);
```

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

Pro náročnější práci s permutacemi či jinými kombinatorickými strukturami musíme použít balíky. V případě permutací lze použít v případě rozkladu na součin nezávislých cyklů příkaz `convert s` parametrem `disjcyc`. Permutaci zadáváme ve tvaru seznamu, kde i-tý prvek seznamu je číslo, na které se zobrazí číslo i (pokud se nejedná o permutaci, dostaneme

chybové hlášení). Výstup již však dostaneme ve tvaru seznamů, kde jednotlivé vnitřní seznamy odpovídají cyklům:

```
> convert([4,6,5,1,7,2,3],disjcyc);
```

$$[[1, 4], [2, 6], [3, 5, 7]]$$

Pro skládání a inverze permutací však již musíme použít balík `group` pokrývající problematiku teorie grup. Konkrétně použijeme příkaz `invperm` pro inverzi permutace (zde ji musíme zadat ve tvaru nezávislých cyklů, proto se nám bude hodit předchozí výsledek) nebo příkaz `mulperms` pro skládání permutací:

```
> group[invperm](%);
```

$$[[1, 4], [2, 6], [3, 7, 5]]$$

```
> group[mulperms]([[4,6,5,1,7,2,3]],[[7,1,4,3,6,5,2]]);
```

$$[[2, 6], [4, 5]]$$

Jak je vidět, dostali jsme i v posledním případě výsledek ve tvaru nezávislých cyklů.

Dalším důležitějším k rozšíření předchozích možností je balík `combinat`. Ten nejen rozšiřuje možnosti předchozích příkazů, ale umí jednotlivé možnosti i generovat. Například všechny podmnožiny dané množiny obdržíme jedním z příkazů `choose` nebo `powerset`:

```
> with(combinat):
Warning, the protected name Chi has been redefined and unpro-
tected
> choose({a,b,c,d});
```

$$\{\{\}, \{b, c, d\}, \{c, d\}, \{a, c, d\}, \{d\}, \{a, d\}, \{b, d\}, \{a, b, d\}, \{a\}, \{b\}, \{a, b\}, \{c\}, \{a, c\}, \{b, c\}, \{a, b, c\}, \{a, b, c, d\}\}$$

```
> powerset({a,b,c});
```

$$\{\{\}, \{a\}, \{b\}, \{a, b\}, \{c\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$$

Podobně příkazem `permute` se dvěma parametry `n,k` obdržíme všechny `k`-prvkové podseznamy seznamu {1,…,n} (tedy zde záleží na pořadí). Zajímá-li nás jen jejich počet (který odpovídá počtu variací bez opakování), stačí použít příkaz `numbperm` s týmiž parametry (zde můžeme dokonce druhý parametr vynechat a získat tak další variantu faktoriálu):

```
> numbperm(5);
```

$$120$$

```
> numbperm(5,2);
```

$$20$$

```
> permute(5,2);
```

$$[[1, 2], [1, 3], [1, 4], [1, 5], [2, 1], [2, 3], [2, 4], [2, 5], [3, 1], [3, 2], [3, 4], [3, 5], [4, 1], [4, 2], [4, 3], [4, 5], [5, 1], [5, 2], [5, 3], [5, 4]]$$

Možnosti funkce pro binomické koeficienty ze standardní knihovny rozšiřuje příkaz pro multinomické koeficienty, který se v balíku jmenuje `multinomial`. Není omezen ani ve velikosti čísel, ani v počtu argumentů.

```
> multinomial(16,4,5,7);
```

$$1441440$$

Posledním důležitým a často využívaným příkazem je `fibonacci`, který k danému celému (tedy i zápornému) číslu najde Fibonacciho číslo s příslušným indexem. Několik prvních Fibonacciho čísel tedy získáme s pomocí příkazu `seq` pro vytvoření posloupnosti, viz nápověda (`?seq`):

```
> seq(fibonacci(i),i=1..20);
```

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765$$

## 6.2 Teorie čísel

Otázky týkající se prvočíselnosti a dělitelnosti mají bezprostřední dopad například na kryptografické aplikace. Ačkoliv Maple je jazykem vysoké úrovně a není tudíž příliš rychlý pro tyto účely, může nám však výrazně pomoci při výzkumu vlastností některých čísel.

Základním algoritmem pro teorii čísel je počítání největšího společného dělitele pomocí příkazu `gcd` a nejmenšího společného násobku pomocí příkazu `lcm`:

```
> gcd(31368,21768);
```

$$24$$

```
> lcm(120,105);
```

$$840$$

Rozklad na prvočinitele zajišťuje příkaz `ifactor` zmíněný již ve druhé kapitole. Opět zde nejsme omezeni velikostí čísla, ale pouze časem, který jsme ochotni věnovat výpočtu (jde jak známo o velmi těžký problém, na kterém stojí většina kryptografických aplikací).

```
> ifactor(840);
```

$$(2)^3 (3) (5) (7)$$

```
> ifactor(2^67-1);
```

$$(7618382257287) (193707721)$$

Tohoto příkazu a pravděpodobnostního ověřování prvočíselnosti využívá řada příkazů týkajících se prvočísel. Uveďme alespoň některé:

- `isprime` – test na prvočíselnost (vrací logickou hodnotu)
- `ithprime` – i-té prvočíslo v pořadí
- `nextprime` – nejbližší větší prvočíslo
- `prevprime` – nejbližší menší prvočíslo

Demonstrujme je na příkladech:

```
> isprime(2^67-1);
```

$$false$$

```
> isprime(8933);
```

```
                                true
> nextprime(10000);
                               10007
> prevprime(10000);
                                9973
> ithprime(1000);
                                7919
```

Další rozšíření (které zde nebudeme uvádět a odkážeme čtenáře na nápovědu) poskytuje balík `numtheory`. Ten umí nejen čísla faktorizovat, ale generovat i všechny dělitele nebo počítat největšího společného dělitele i komplexních čísel. Následuje přehled nejjednodušších příkazů z tohoto balíku:

- `bigomega` – počet prvočíselných faktorů včetně násobnosti
- `divisors` – množina všech dělitelů
- `factorset` – množina všech prvočíselných faktorů
- `GIgcd` – největší společný dělitel komplexních čísel
- `pi` – počet prvočísel menších nebo rovných danému číslu (pozor na podobnost s Ludolfovým číslem, Maple uvažuje malá a velká písmena!)

Příklady použití těchto funkcí:

```
> with(numtheory):
Warning, the protected name order has been redefined and un-
protected

> bigomega(8400);
                                   8
> divisors(8400);
 {1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 15, 16, 20, 21, 24, 25, 28, 30, 35, 40, 42, 48, 50, 56, 60,
      70, 75, 80, 84, 100, 105, 112, 120, 140, 150, 168, 175, 200, 210, 240, 280, 300,
      336, 350, 400, 420, 525, 560, 600, 700, 840, 1050, 1200, 1400, 1680, 2100, 2800,
      4200,
      8400}
> factorset(8400);
                               {2, 3, 5, 7}
> GIgcd(-345+515*I,1574+368*I);
                              41 + 117 I
> pi(1000);
                                  168
```

# Kapitola 7

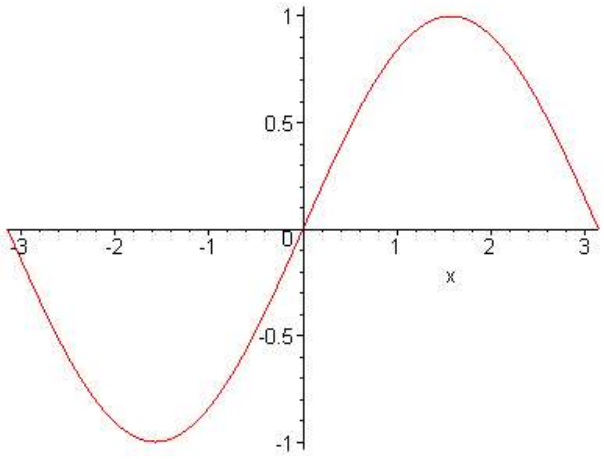
## Grafické možnosti systému Maple

V této kapitole se seznámíme s možnostmi grafické vizualizace výsledků v Maplu, a to jak funkcí jedné proměnné, tak i funkcí dvou proměnných. Budeme se zabývat i vizualizací funkcí implicitně zadaných, slučováním grafů do jednoho grafu, animacemi a na závěr vizualizací řešení diferenciálních rovnic.

### 7.1 Dvourozměrné grafy

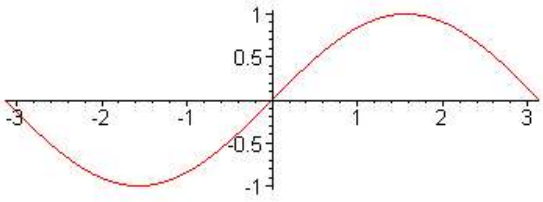
Základním příkazem pro vytvoření dvourozměrných grafů je příkaz `plot`. V jeho nejjednodušší variantě mu stačí zadat funkci nebi i výraz, jejichž graf má vytvořit, a interval nezávisle proměnné, na který se má graf omezit (měřítko na ose závisle proměnné je pak automaticky zvoleno tak, aby se všechny hodnoty funkce na daném intervalu podařilo zobrazit).

```
> plot(sin(x),x=-Pi..Pi);
```

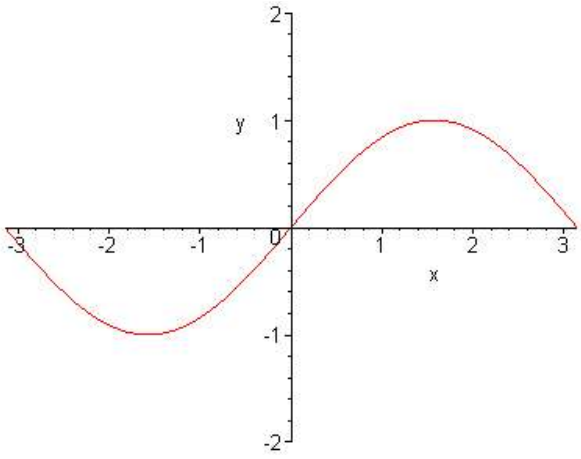



Pokud chceme zachovat stejná měřítka na osách, můžeme použít jednu ze široké nabídky voleb – jako další parametr zadáme `scaling=constrained`. Případně lze jakkoliv změnit poměr os tím, že specifikujeme rozsah i na ose y:

```
> plot(sin,-Pi..Pi,scaling=constrained);
```



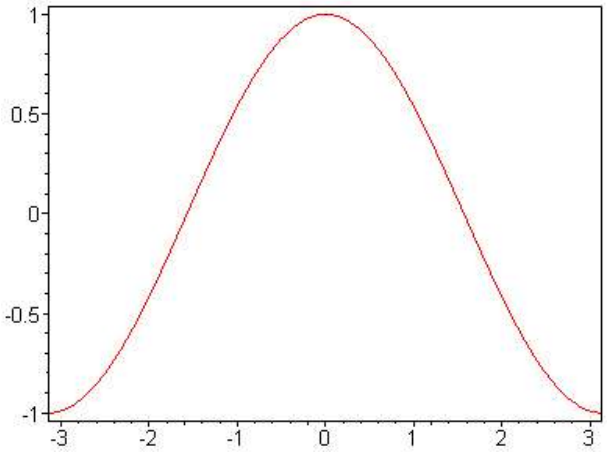

```
> plot(sin(x),x=-Pi..Pi,y=-2..2);
```




Nyní uvedeme přehled nejdůležitějších parametrů příkazu `plot`. Jejich celkový přehled nalezneme v nápovědě příkazem `?plot[options]`.

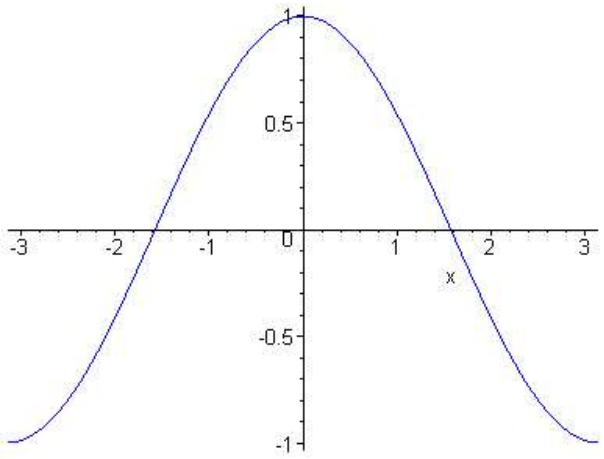
Pomocí parametru `axes=...` zadáváme zobrazení os. Chceme-li osy zobrazit jiným způsobem než v základní variantě, můžeme zvolit ještě mezi rámem (osa x je dole a y vlevo), boxem (obrázek je ohraničen osami ze všech čtyř stran) nebo zobrazením bez os:

```
> plot(cos,-Pi..Pi,axes=BOXED);
```



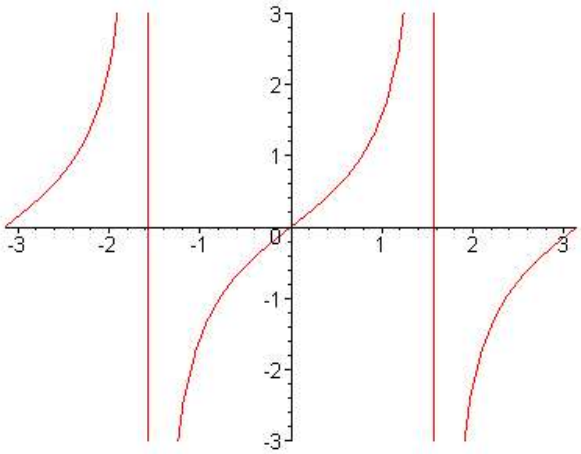

Pomocí parametru `color=...` určujeme barvu, kterou se má graf vykreslit. Barvu lze zadat buďto z nabídky základních barev slovním popisem (`blue, red, green` apod.) nebo její RGB hodnotou. Maple akceptuje tento parametr i s variantou `colour=...` se stejným významem.

```
> plot(cos(x),x=-Pi..Pi,color=blue);
```

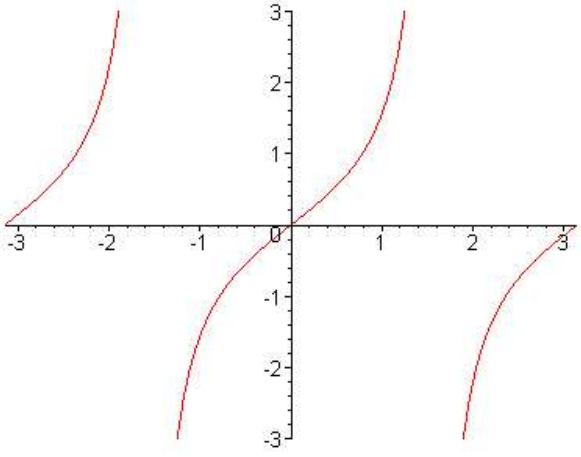



Pro nespojitou funkci je vhodné použít již dříve zmíněný parametr `discont=...` Zadáme-li jeho hodnotu `true`, pak Maple nejprve nalezne body nespojitosti funkce. V opačném případě Maple nakládá se zadanou funkcí jako se spojitou a v bodech nespojitosti se pak objeví svislé asymptoty, které ovšem ve skutečnosti nejsou asymptotami, ale spojnicemi nejbližšího bodu vlevo od nespojitosti a nejbližšího bodu vpravo od nespojitosti.Ukážeme to na příkladu funkce tangens.:

```
> plot(tan,-Pi..Pi,-3..3,discont=false);
```



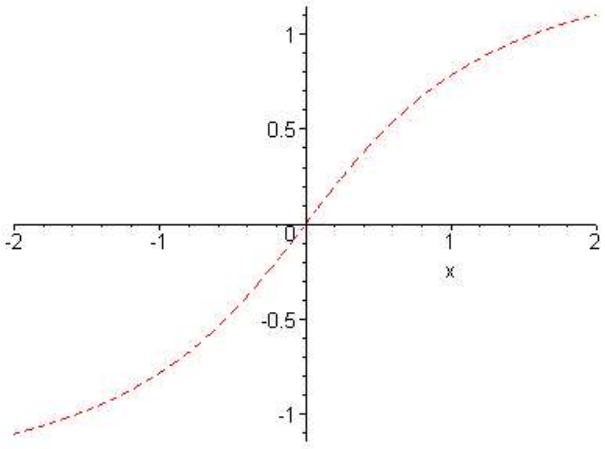

```
> plot(tan,-Pi..Pi,-3..3,discont=true);
```




Pomocí parametru `linestyle=...` měníme styl křivky, kterou je graf vykreslen. Tato volba poskytuje kromě barvy další možnost rozlišení jednotlivých křivek v grafu (k

zobrazování více křivek v jednom grafu se dostaneme později). Maple umožňuje zvolit kromě plné čáry také čáru tečkovanou, čárkovanou nebo čerchovanou.

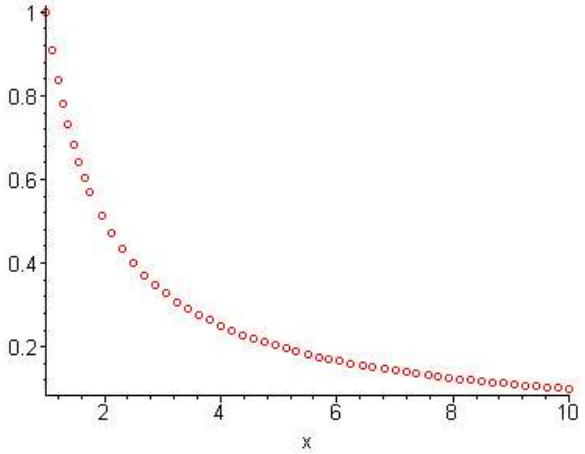
```
> plot(arctan(x),x=-2..2,linestyle=DASHDOT);
```




Pomocí parametru `numpoints=...` volíme v Maplu kolik funkčních hodnot se má vypočítat při tvorbě grafu. Maple pak rozdělí zadaný interval na příslušný počet dílků, v jejichž krajních bodech počítá funkční hodnoty. Větší hodnotou tohoto parametru docílíme lepšího výsledku, ale prodloužíme dobu výpočtu.

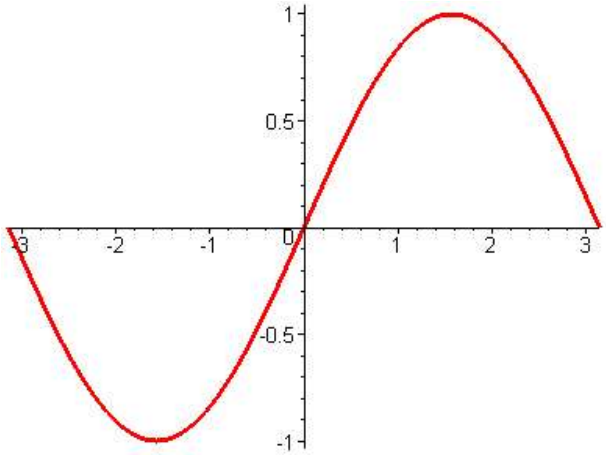
Chceme-li zobrazit graf funkce pouze v některých bodech (či jiným způsobem), spojíme dva parametry: parametr `style=point,` kterým předepisuje, že zobrazujeme pouze body (tato volba má více variant, které najdete v nápovědě) a parametr `symbol=...,` kterým volíme použitý symbol (kroužek, čtvereček apod.). Podobný efekt nabízí příkaz `pointplot`. Ten má širší použití – lze mu zadat, ve kterých bodech chceme hodnotu zobrazit.

```
> plot(1/x,x=1..10,style=POINT,symbol=CIRCLE);
```



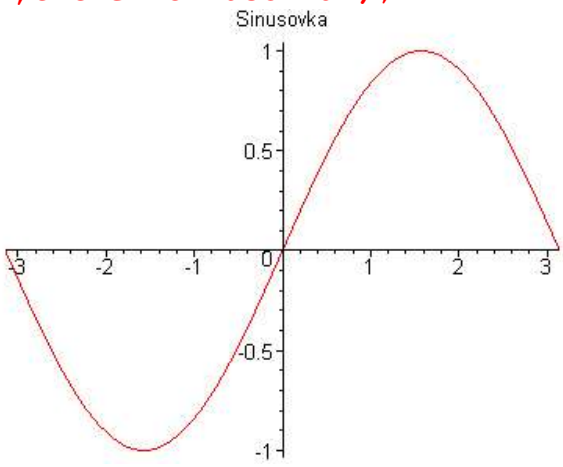

Tloušťku čar nastavíme pomocí parametru `thickness=...,` kde mu přiřadíme celé číslo mezi 0 a 3 (3 znamená největší tloušťku):

```
> plot(sin,-Pi..Pi,thickness=3);
```



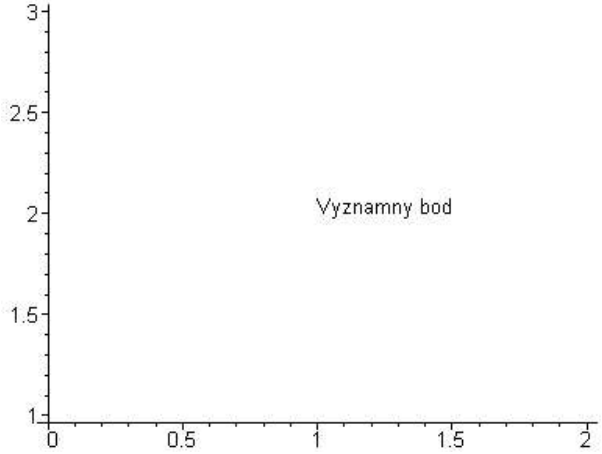

Titulek grafu lze nastavit pomocí parametru `title=...`, jehož argumentem je textový řetězec. Tento řetězec se pak zobrazí do horní části grafu:

```
> plot(sin,-Pi..Pi,title="Sinusovka");
```




Pro další popis grafu lze použít parametr `TEXT` nebo příkaz `textplot z` balíku `plots`. Takto lze vytvořit obrázek složený pouze z textu, který pak pomocí skládání grafů můžeme sloučit s jiným grafem a získat tak text v libovolném místě grafu. Významným parametrem těchto příkazů je `align`, který specifikuje polohu textu vzhledem k zadané souřadnici.

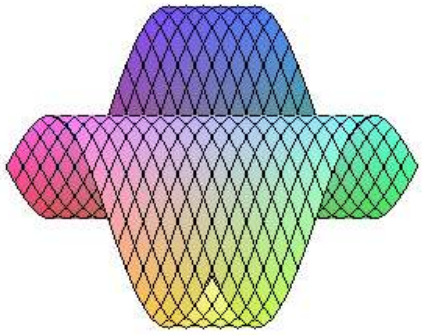
```
> plots[textplot]([1,2,`Vyznamny bod`],align={ABOVE,RIGHT});
```



## 7.2 Třírozměrné grafy

Pro trojrozměrné grafy se v Maple používá příkaz `plot3d`, který zobrazuje grafy funkcí dvou proměnných, proto musíme specifikovat jeho rozsah na dvou osách nezávisle proměnných:

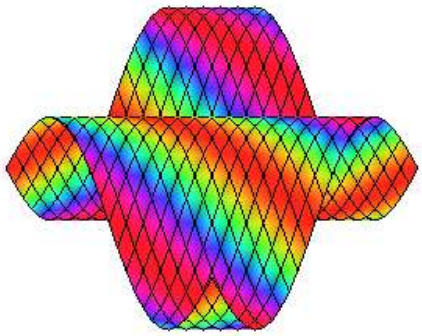
```
> plot3d(sin(x+y),x=-Pi..Pi,y=-Pi..Pi);
```



Parametry příkazu `plot3d` jsou ve většině případů shodné s parametry příkazu `plot` a jejich podrobný přehled najdeme vyvoláním nápovědy `?plot3d[option]`. Zásadní odlišnosti pro příkaz `plot3d` uvedeme ve zbytku této podkapitoly.

Barva může být specifikována nejen fixním slovem nebo pomocí tzv. RGB palety, ale také jako procedura či funkce dvou proměnných – souřadnic. Toho lze využít pro dosažení kvalitních grafických výstupů a pro zkušenější uživatele lze touto cestou dojít například i ke kreslení fraktálů:

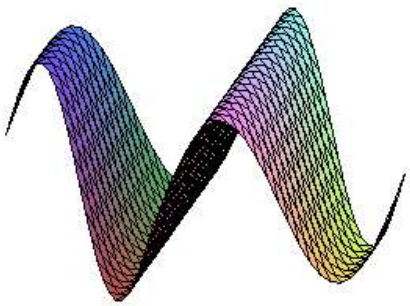
```
> plot3d(sin(x+y),x=-Pi..Pi,y=-Pi..Pi,color=cos(2*x+y));
```



Místo parametru `numpoints`, se používá parametr `grid` s dvěma dalšími subparametry, kterými udáváme hustotu sítě, ve které se počítají funkční hodnoty, na obou osách. Délka výpočtu je pak úměrná součinu obou hodnot.

Chceme-li změnit náhled na obrázek, je možno využít parametr `orientation`, kterým zadáme dva úhly, pod kterými se chceme na graf dívat. Hodnoty úhlů zadáváme ve stupních a mohou být i záporné.

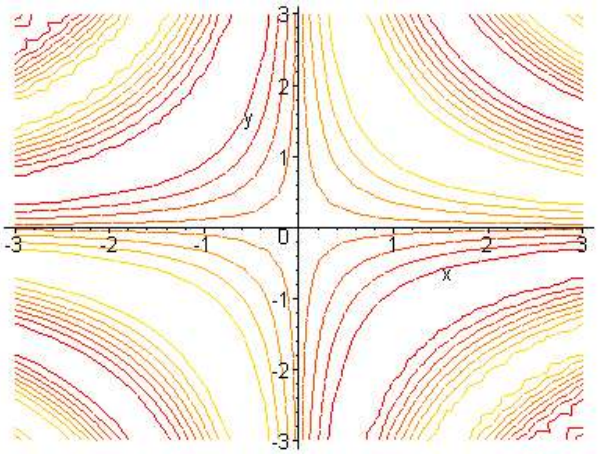
```
> plot3d(sin(x+y),x=-Pi..Pi,y=-Pi..Pi,orientation=[-30,60]);
```



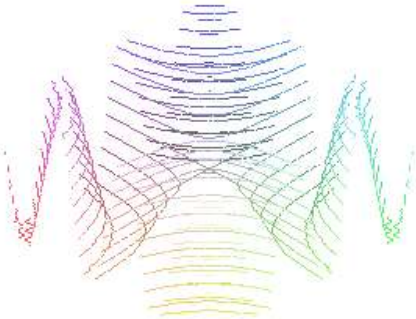
Rozšířena je i nabídka stylů, jejich přehled ovšem ponecháme uživateli k nastudování v nápovědě.

Pro funkce více proměnných lze kromě jejich grafů kreslit také jejich vrstevnice, které můžeme získat příkazem `contourplot` nebo `contourplot3d`.

```
> plots[contourplot](sin(x*y),x=-3..3,y=-3..3);
```




```
> plots[contourplot3d](sin(x*y),x=-Pi..Pi,y=-Pi..Pi);
```

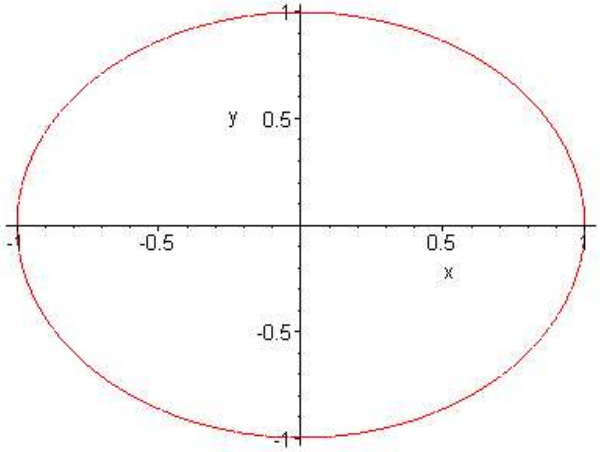


Dále Maple umožňuje zobrazovat třírozměrné grafy také v jiných souřadných soustavách (polární, sférická, atd.). K nastudování této problematiky doporučujeme nápovědu.

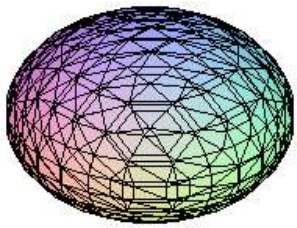
## 7.3 Grafy zadané implicitně

Maple poskytuje dobré grafické možnosti i v případě, kdy znázorňovaná funkce nebo výraz nejsou zadány explicitně vzhledem k nezávisle proměnné nebo proměnným, ale je definovaná implicitně např. rovnicí. V těchto případech využijeme z balíku `plots` resp. `plots3d` příkaz `implicitplot`, resp. `implicitplot3d`. Jejich nepovinné parametry jsou shodné s výše uvedenými příkazy `plot` a `plot3d`, nesmíme ovšem zapomenout na to, že musíme specifikovat rozsah grafu ve všech souřadných osách a že kvalitu zobrazení grafu ovlivníme ve dvourozměrném případě parametrem `grid`.

```
> with(plots):
Warning, the name changecoords has been redefined

> implicitplot(x^2+y^2=1,x=-1..1,y=-1..1);
```



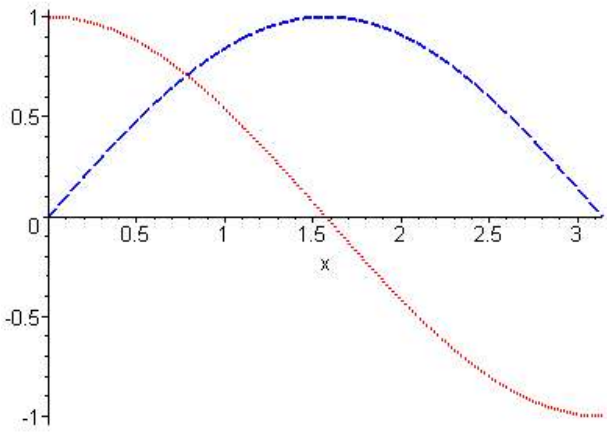

```
> implicitplot3d(x^2+y^2+z^2=1,x=-1..1,y=-1..1,z=-1..1);
```



## 7.4 Slučování grafů do jednoho grafu

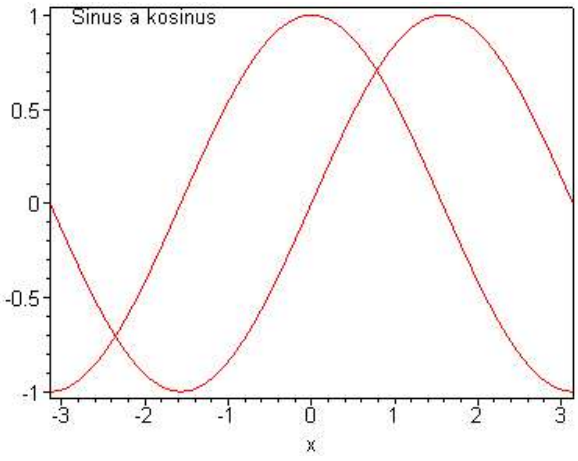
Jestliže již umíme vytvářet grafy funkcí, je často vhodné umístit více grafů funkcí do jednoho grafu. V Maplu máme více možností, jak to provést. V příkazu `plot` lze zadat seznam nebo množinu funkcí, které se zobrazí do jednoho grafu a odpovídající parametry musíme pak též zadat ve formě seznamů parametrů:

```
> plot([cos(x),sin(x)],x=0..Pi,linestyle=[2,3],color=
[red,blue],thickness=[2,2]);
```




Širší možnosti ovšem nabízí příkaz `display` z balíku `plots`. Nejen, že zde může být jedním z grafů například text či jakákoliv jiná grafická struktura, ale tímto příkazem můžeme do jisté míry nahradit i animaci, jak uvidíme v následující podkapitole. Parametry grafu stačí zadat jednou, není třeba je zadávat v seznamech jako v předchozím případě. Pokud jsou některé z parametrů specifické jen pro některý z grafů, stačí je zadat při jeho vytvoření.

```
> with(plots):
Warning, the name changecoords has been redefined

> P1:=plot(sin(x),x=-Pi..Pi):
> P2:=plot(cos(x),x=-Pi..Pi):
> P3:=textplot([-2,1,`Sinus a kosinus`]):
> display({P1,P2,P3},axes=boxed);
```

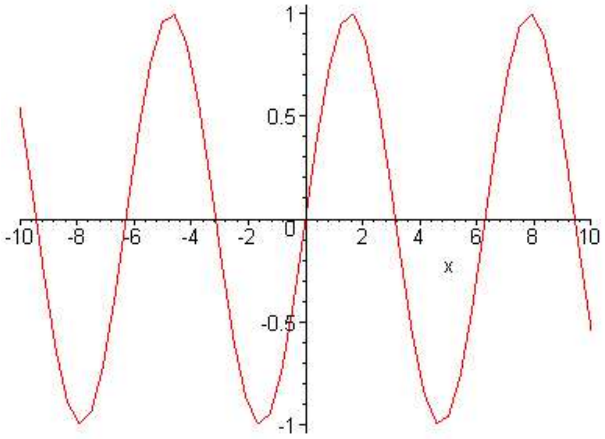



## 7.5 Animace

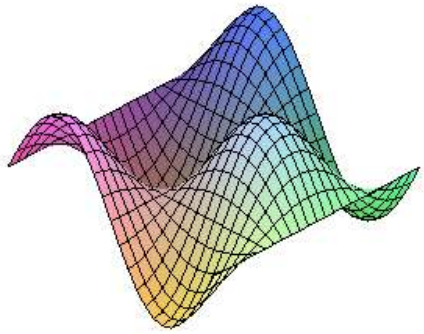
Maple dokáže grafy nejen kreslit, ale umožňuje také jednoduchou animaci (posloupnost zobrazení („snímků“) grafu v závislosti na animované proměnné). K tomu slouží příkazy `animate`, resp. `animate3d` z balíku `plots`. Jejich parametry jsou shodné s ostatními výše uvedenými příkazy `plot` a `plot3d`, navíc zde můžeme specifikovat počet snímků grafu pomocí parametru `frames`. Nezadáme-li osy grafu, budou ve třírozměrném případě pro jeho větší přehlednost vynechány. Jako první parametr u těchto příkazů se zadává funkce nezávisle proměnných, z nichž jednou je čas (obvykle označen jako t). Ukážeme to na následujících příkladech:

```
> with(plots):
Warning, the name changecoords has been redefined
```

```
> animate(sin(x*t),x=-10..10,t=1..2,frames=50);
```

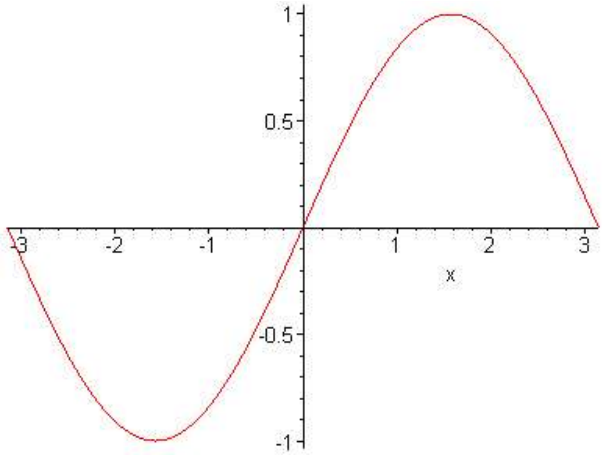



```
> animate3d(cos(t*x)*sin(t*y),x=-Pi..Pi, y=-Pi..Pi,t=1..2);
```



Druhou možností animace je použít příkazu `display` s parametrem `insequence=true`. Tento postup lze doporučit v případě, kdy funkce není spojitě závislá na čase(nezávisle proměnné animace) – například takto lze graficky demonstrovat některé iterační postupy. Jednotlivé grafy ze zadaného seznamu se pak berou jako jednotlivé snímky animace. Následující postup získání animovaného obrázku se zdá být poněkud krkolomnější než předchozí, nicméně v některých situacích dostaneme daleko lepší výsledky než klasickou animací:

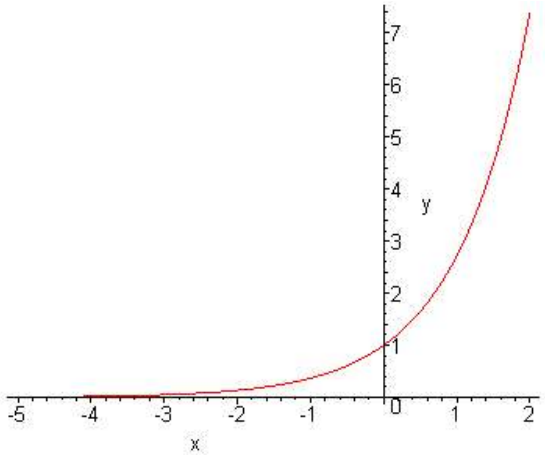
```
> for i from 1 to 10 do
> p[i]:=plot(sin(i*x),x=-Pi..Pi):
> od:
> display([seq(p[i],i=1..10)],insequence=true);
```




## 7.6 Grafické znázornění řešení diferenciálních rovnic

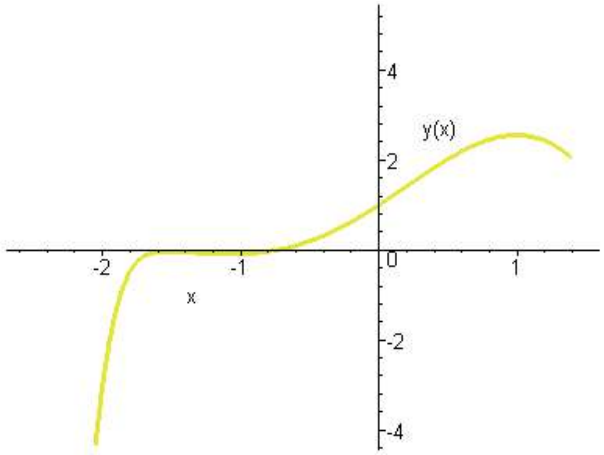
Speciální grafy v Maplu tvoří řešení diferenciálních rovnic. Pro tyto účely jsou v Maplu zavedeny speciální příkazy, jako např. příkaz `odeplot` z balíku `plots`. Jeho parametrem je řešení diferenciální rovnice, které se zde uvažuje numericky, neboť při zobrazení grafu je nutno mít všechny hodnoty vyčísleny numericky. Použití příkazu `odeplot` opět ukážeme na příkladu:

```
> p:= dsolve({D(y)(x) = y(x), y(0)=1}, type=numeric, range=-
5..2):
> odeplot(p);
```




Příkaz `odeplot` umožňuje zobrazit pouze numerická řešení diferenciálních rovnic. Maple 9.5 však obsahuje další balíky `DEtools`, resp. `PDEtools` a v nich příkazy `DEplot`, resp. `PDEplot`, které tento nedostatek odstraňují. V parametrech těchto příkazů zadáváme totiž přímo příslušnou diferenciální rovnici s počátečními (okrajovými) podmínkami. Význam jednotlivých parametrů opět naleznete v nápovědě. Ukážeme to na jednoduchých příkladech:

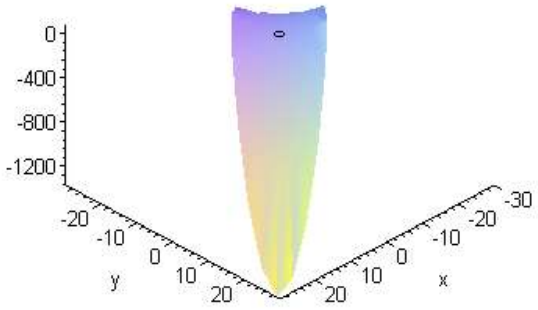
```
> with(DEtools):
> DEplot (cos(x) * diff(y(x),x$3) - diff(y(x),x$2) + Pi * diff
(y(x),x) = y(x) - x,y(x), x=-2.5..1.4, [ [y(0)=1, D(y)(0)=2,
(D@@2)(y)(0)=1] ], y=-4..5, stepsize=.05);
```




```
> with(PDEtools):
> pde1 := diff(u(x,y),x)*diff(u(x,y),y)-x*y+u(x,y)=0;
```

$$pde1 := \left(\frac{\partial}{\partial x}u(x,y)\right)\left(\frac{\partial}{\partial y}u(x,y)\right) - x\,y + u(x,y) = 0$$

```
> PDEplot(pde1, [cos(t),sin(t),0], t=-Pi..Pi,
ic_assumptions=[diff(u(x,y),x) = -cos(t)]);
```



# Kapitola 8

## Grafické aplikace – technologie Maplet

Od verze 7 je v Maplu zaveden balík `Maplets`, který umožňuje vytvořit interaktivní grafické uživatelské prostředí, které nazveme „maplety“, jejichž prostřednictvím lze snadněji přistupovat k maplovským aplikacím. V Maple 9.5 fungují maplety na stejném principu jako javovské aplety v internetových prohlížecích.

Tyto maplety pak lze spouštět buď přímo v Maplu nebo nezávisle na Maplu pomocí programu Maplet Viewer. Přitom lze využívat všech možností systému Maple, vytvářet grafy funkcí, nebo otevírat další okna.

K základní výuce mapletů slouží přehledný systém zápisníků, tzv. „Roadmap“. Jde o velmi dobře zpracovanou učebnici, která obsahuje širokou sadu příkladů. Roadmap lze vyvolat příkazem

```
> ?roadmap;
```

Pro maplety je také připravena rozsáhlá dokumentace, která obsahuje velké množství příkladů, které je možné dále využít a upravovat. Dokumentaci lze vyvolat pomocí příkazu

```
> ?Maplets;
```

### 8.1 Vytváření Mapletů

Pomocí mapletů lze vytvářet komunikační rozhraní, které načítá uživatelův vstup, zpracovává jej a prezentuje výsledky. Tento cyklus lze libovolně opakovat, případně vytvořit výpočet, který vytváří další navazující výsledky. V rámci mapletu je tak možno například načíst rovnici, vyřešit ji a vypsat její výsledek, v dalším kroku pak zobrazit graf původní funkce spolu s řešením.

Maplet je sada vzájemně propojených prvků, prezentačních oken, dialogových oken a navázaných akcí. Maplety se vytváří ve dvou fázích: v první je vytvořen objekt mapletu a ve druhé je daný maplet zobrazen.

Základním krokem pro vytváření Mapletů je inicializace balíku `Maplets` a jeho příkazu `Elements`:

```
> with(Maplets): with(Maplets[Elements]):
```

Vlastní objekt mapletu je tvořen popisem vlastností mapletu a víceúrovňovým seznamem jednotlivých vnořených elementů:

```
> Maplet (Window ('title'="Maplet",
     [Element1, Element2, ...] ) ):
```

Jednoduchá aplikace může pak vypadat takto:

```
> with (Maplets [Elements]):
> with (Maplets):
> maplet1 := Maplet( Window( 'title'="Maplet", [
    ["Integrovany vyraz: ", TextField ['TF1'] () ],
    ["Promenna integrace: ", TextField ['TF2'] (3) ],
    TextBox ['TB1'] ('editable' = 'false', 3..40 ),
    [Button ("Integruj", Evaluate ('TB1'='int(TF1, TF2)')),
     Button ("OK", Shutdown ( ['TF1', 'TF2', 'TB1'] ) ),
```

```
        Button ("Vymaz", SetOption ('TF1' = "") ) ]
    ] ) ):
> Display ( maplet1 );
```

Výše uvedený maplet se skládá z těchto částí:

- Dvou vstupních textových polí – první a druhá položka seznamu. Jde opět o seznamy – tímto způsobem jsou prvky umístěny do řádku (více o lokalizaci elementů viz kapitola 8.2). První položka obsahuje zobrazený textový popis, druhá položka jednoduchý „konstruktor“ (`TextField`) vlastního textového pole. Parametr v hranatých závorkách (`TF1, TF2`) je identifikátor daného pole, v kulatých závorkách jsou uvedeny další parametry, například délka pole.
- Textové pole určené pro víceřádkový výstup (`TextBox`). Opět obsahuje identifikátor a další parametry popisující například rozměry pole.


- Poslední část tvoří objekty pro tlačítka. Každé tlačítko má jednak svůj popis („Integruj“, „OK“, „Vymaž“) a dále funkci, která je provedena při jeho zmáčknutí. Tato funkce může pro svůj běh využit hodnoty z některých polí pomocí jejich identifikátorů.
  Některé příkazy jsou již předdefinované (`Shutdown, SetOption`), pro spouštění externích příkazů je zapotřebí využít rozhraní příkazu `Evaluate`.

Na posledním řádku je spuštění objektu mapletu, ukázka práce s ním je na následujícím obrázku:

Funkce `Display` vrací také hodnoty objektů uvedených jako parametry příkazu `Shutdown.`

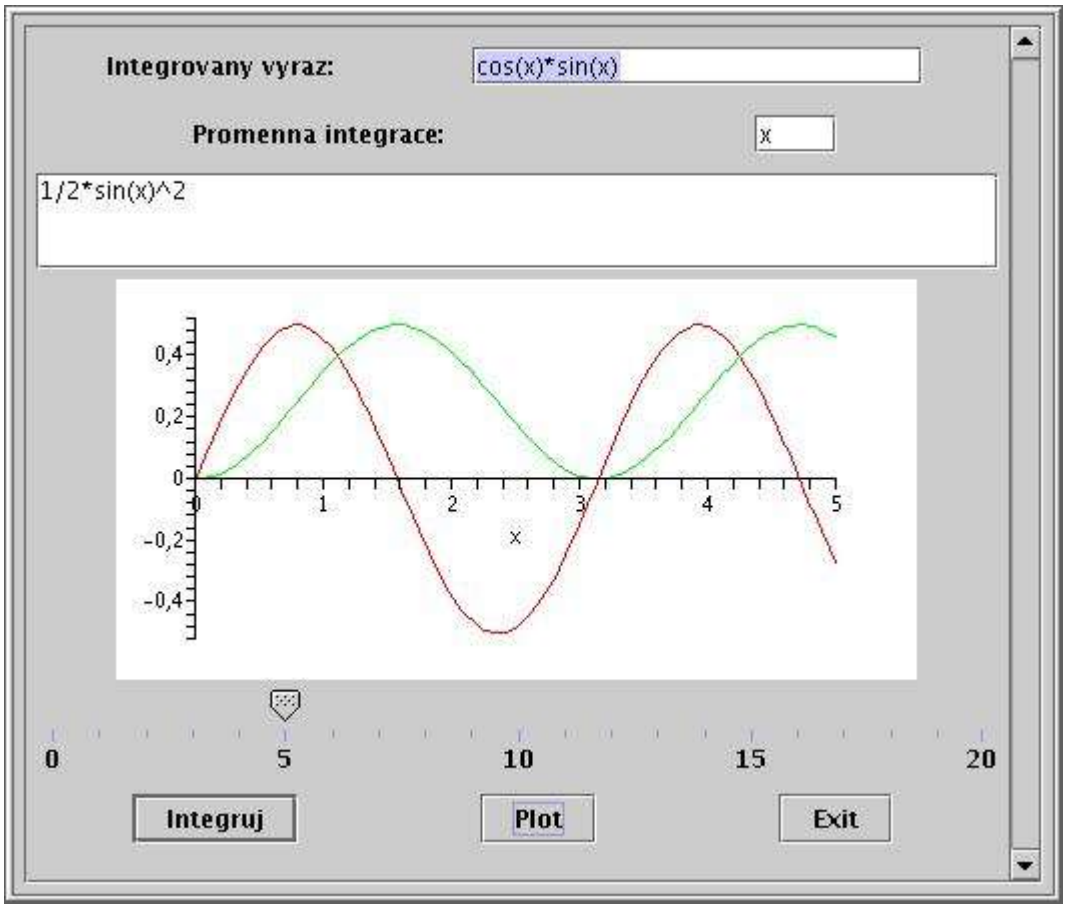
*["cos(x)*sin(x)", "x", "-1/2*cos(x)^2"]*

## 8.2 Náročnější aplikace

Další možností, kterou si ukážeme, je vkládání grafických elementů domapletu, konkrétně grafů. K tomu nám poslouží element `Plotter`. Dalším důležitým elementem je `Slide`, pomocí kterého se vytváří lišta umožňující měnit rozlišení grafu. Jeho parametry pak určují například rozsah lišty, implicitní hodnotu či prvky na liště.

```
> maplet2 := Maplet (Window (BoxColumn( 'vscroll'='always',
    ["Integrovany vyraz: ", TextField['TF1']()],
    ["Promenna integrace: ", TextField['TF2'](3)],
    TextBox['TB1']( 'editable' = 'false', 3..40 ),
    Plotter['PL1'](),
```

```
    Slider['SL1'] (0..20, 5, 'showticks', 'majorticks'=5,
     'minorticks'=1, 'visible'='false',
    Evaluate( 'PL1' = 'plot ([TF1, TB1], TF2=0..SL1)' ) ),
    [Button ("Integruj", Action (Evaluate
      ('TB1'='int(TF1, TF2)'),
       SetOption ('B1'(enabled)='true') )),
    Button['B1'] ("Plot", 'enabled'='false',
    Action (SetOption ('SL1'('visible')='true'), Evaluate
       ('PL1'='plot ([TF1, TB1], TF2=0..SL1)' ) ) ),
    Button ("Exit", Shutdown(['TF1', 'TF2', 'TB1']))]
) ) ):
> Display (maplet2);
```




Provádění tohoto mapletu je dvoukrokové. V prvním kroku je vypočítána a zobrazena hodnota integrandu (tlačítkem `Integruj`) , ve druhém je vykreslen graf původní a integrované funkce (tlačítkem `Plot`). Při tomto kroku je také zobrazena lišta pro změnu rozlišení. Výsledný maplet vypadá takto:

V dalším uvedeme přehled funkcí, které je možné využít v rámci mapletů. U vybraných oblastí se zaměříme také na způsob jejich případné použití.

### 8.2.1 Elementy okna mapletu

Jde o základní stavební vizuální prvky okna mapletu, které tvoří jeho design. Použití některých z nich (`TextField, Button, Slider`)

```
Button      CheckBox     ComboBox    DropDownBox     Label
ListBox     Plotter      RadioButton    Slider       Table
TextBox     TextField    ToggleButton
```

### 8.2.2 Dialogová okna

Maplety mohou vyvolávat také dialogová okna, která ovšem mají -- na rozdíl od hlavního okna – pevně definovaný vzhled a chování a nelze vkládat další prvky.

```
Alert Dialog    Color Dialog    Confirm Dialog    File Dialog
Input Dialog    Message Dialog  Question Dialog
```

### 8.3.3 Použití menu

Okno mapletu může obsahovat menu a v nich položky a oddělovače. Použití menu je naznačeno v tomto příkladě:

```
> maplet3 := Maplet(
    Window ('title'="Integral a derivace", 'menubar'='MB1',
      ["Vloz vyraz a vyber polozku z menu:",
      [TextField ['TF1'] () ],
      Button ("Exit", Shutdown (['TF1']) ) ] ),
      MenuBar ['MB1'](
        Menu ("Soubor",
          MenuItem("Close", Shutdown (['TF1']) ) ),
        Menu ("Prikazy",
          MenuItem ("Integral",
            Evaluate ('TF1' = 'int(TF1, x)') ),
          MenuSeparator(),
            MenuItem ("Derivace",
              Evaluate ('TF1' = 'diff(TF1, x)') )
        ) ) ):
> Display (maplet3);
```

### 8.3.4 Prováděcí příkazy

Základní příkazem je v tomto případě příkaz `Action`, který může obsahovat libovolné množství prováděcích příkazů. Tyto příkazy se při spuštění příkazu spouští v zadaném pořadí.

| | | | |
|---|---|---|---|
| `CloseWindow` | `Evaluate` | `RunDialog` | `RunWindow` |
| `SetOption` | `Shutdown` | `Box` | `Grid` |

## 8.3 Rozvržení mapletu

Je třeba dbát nejen o funkčnost mapletu, ale i o jeho vzhled a umístění jednotlivých jeho prvků v rámci okna. Základním postupem je rozdělení jednotlivých prvků do řádků a sloupců.

Nejjednodušším způsobem, jak definovat rozdělení do bloků, je použití vnořených seznamů. Tato jednoduchá možnost je ve své podstatě velmi intuitivní – v každé úrovni zanoření se střídají řádky a sloupce. První úroveň tvoří řádky, další sloupce, další opět řádky … Pro zvýšení přehlednosti je také dobré tuto strukturu udržovat i při zápisu programu, který díky tomu získává na čitelnosti, například takto:

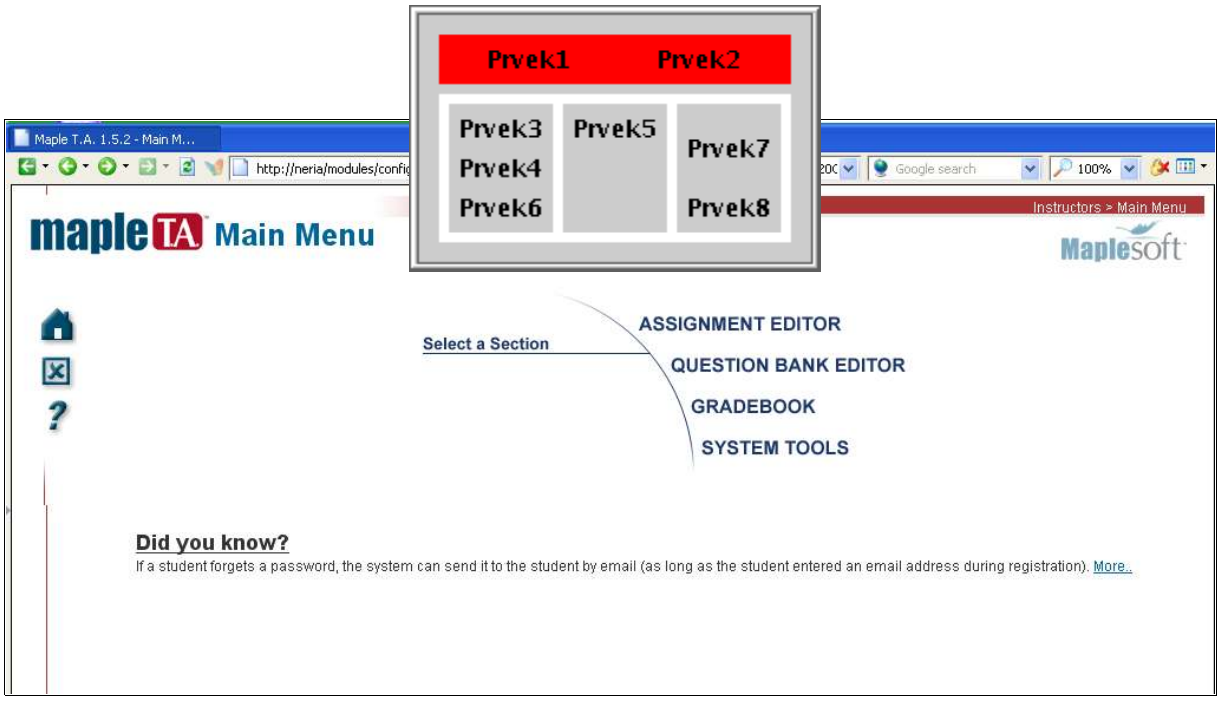
```
> maplet4:= Maplet ( [
    ["Prvek1", "Prvek1a", "Prvek1b", "Prvek1c"],
    ["Prvek2", "Prvek2a"]
  ]):
> Display (maplet3);
```


Tento způsob je vhodný pro jednoduché maplety, použití seznamů má však omezenou čitelnost například v situaci, kdy prvky „přetečou“ délku řádku. V mnoha případech je však zapotřebí organizaci rozvržení prvků zlepšit, chybí možnost změny zarovnání, barvy a dalších vlastností. Prvky navíc leží na řádku zcela náhodně.

Pro tyto účely obsahuje balík `Maplets` také další příkazy, které tyto vlastnosti mají. Jde o příkazy `BoxRow` a `BoxColumn`. Jejich prvky se vkládají do obyčejných kulatých závorek, stejně jako parametry. Výsledkem může být například maplet:

```
> maplet5 := Maplet ( [
     BoxRow (background='yellow', "Prvek1", "Prvek2"),
     BoxRow (background='white',
          BoxColumn (valign='center',
                     "Prvek3", "Prvek4", "Prvek6"),
          BoxColumn (valign='top', "Prvek5"),
          BoxColumn (valign='bottom', "Prvek7", "Prvek8")
          )
     ] ) :
> Maplets [Display] (maplet5);
```

Tento maplet také současně demonstruje nejen možnosti použití kombinace různých barev a různého centrování.




## 8.4 Prohlížeč mapletů

Prvním způsobem práce s maplety je jejich spuštění z Maplu. Zápisník obsahující daný maplet je proveden a zobrazí se okno daného mapletu.

Druhým způsobem je export zápisníku do samostatné aplikace pomocí příkazu `Export` z menu `File`. Výsledkem je soubor s příponou `.maplet`, který lze spustit pomocí programu Maplet Viewer,který je součástí Maple9.5.

Po exportu stačí pouze kliknout na soubor s příponou `.maplet`, kód mapletu se provede a zobrazí se jeho rozhraní. Pochopitelně je zapotřebí, aby na daném počítači byl nainstalován systém Maple 9.5.

# Kapitola 9

## Podpora e-learningu pro systém Maple 9.5

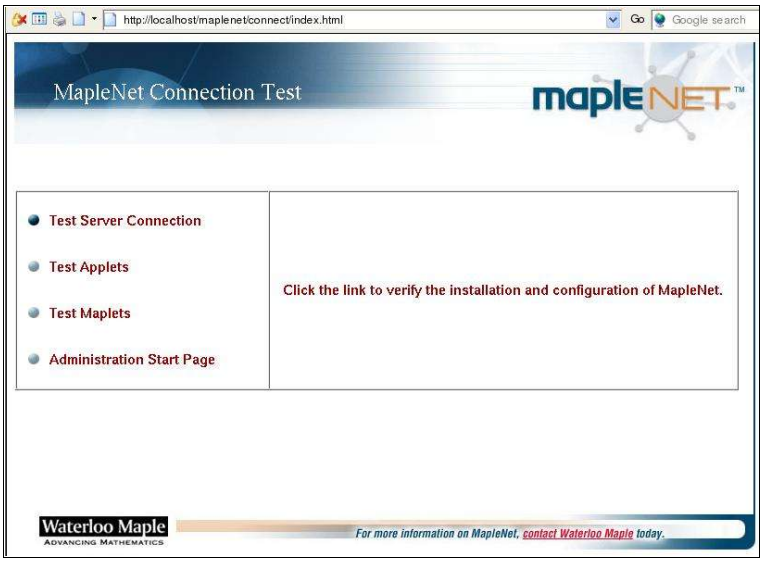
Pro podporu e-learningu matematiky a jejich aplikací vyvinula firma Maplesoft výukový systém LMS (angl. Learning Management System), který sestává z informačních a komunikačních technologií MapleNet a Maple T.A.

Oba tyto systémy jsou napojeny na systém Maple, přičemž systém MapleNet je navázán na již nainstalovaný program Maple, zatímco systém Maple T.A. je kompletní systém obsahující výpočetní nástroje Maple jako svoji součást. Pro práci se systémem MapleNet je dále zapotřebí nainstalovat internetový server – doporučeným systémem je Apache/Tomcat.

### 9.1 MapleNet

Systém MapleNet se skládá ze dvou spolupracujících serverů. První umožňuje klientskému počítači využívat „Mapletovské výukové objekty“ (Maplety a Java MapleNet Aplety) - MVO. Druhý server, takzvaný „publisher“, se využívá jako sklad těchto MVO. Tento systém umožňuje i těm, kteří nemají nainstalovaný Maple, ale mají přístup k serveru MapleNet, aby si maplety zpracovávali prostřednictvím Internetu s využitím Java™ prohlížečů.

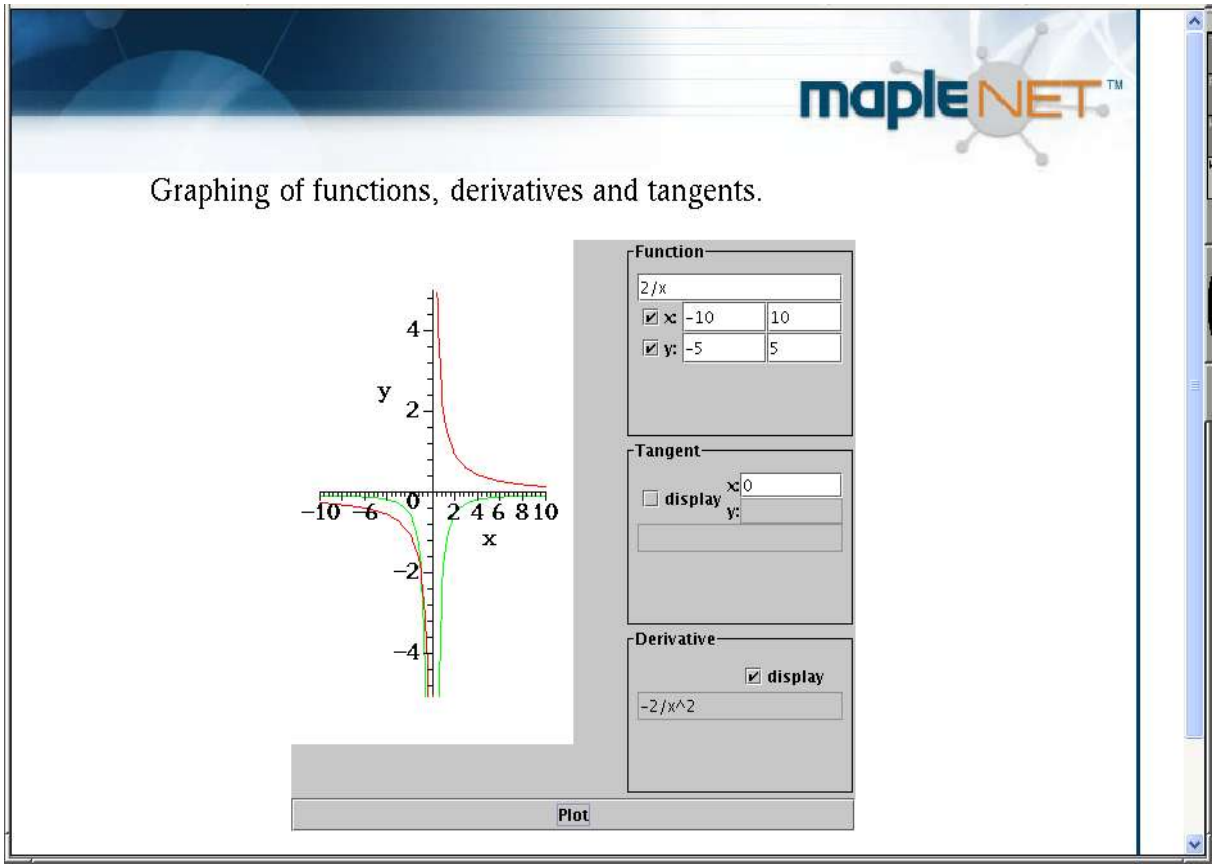
Systém MapleNet lze nainstalovat jak na systém Windows, tak i na systém Linux. Instalace vyžaduje nainstalovaný systém Maple 9.5 a dále Web server – doporučovaný je server Apache ve verzi Jakarta-Tomcat-4, který je volně šiřitelný. Vzhledem k využívání Java apletů je pochopitelně zapotřebí, aby byl na počítači dostupný systém Java ve verzích 1.4 – JDK nebo JRE (konkrétní specifikace je upřesněna v návodu pro instalaci).




*Obrázek č. 9.1: Kontrola funkčnosti MapleNetu*

Po instalaci je zapotřebí spustit server a pomocí nástroje MapleNet Administrator nakonfigurovat celý systém. Konfigurace je jednoduchá a přesně popsaná v nápovědě. Drobnou nevýhodou je neschopnost serveru běžet plně na pozadí. Pro práci se systémem je zapotřebí rovněž spustit Web server (Apache). Systém MapleNet je přístupný pomocí libovolného JSP-kompatibilního prohlížeče (MS Internet Explorer, Opera). Zkontrolovat funkčnost MapleNetu lze pak na testovací obrazovce, viz obr. 9.1.

Druhou důležitou součástí systému MapleNet je server Publisher. Ten funguje jako centrální databáze Java apletů určených k prezentaci pomocí rozhraní MapleNetu. Tyto aplety jsou obdobou mapletů, pouze s tím rozdílem, že jsou spouštěny ze vzdáleného počítače. Mohou tak pracovat bez přítomnosti systému Maple nainstalovaného na klientském počítači, přičemž pro svůj běh využívají vzdálený server MapleNet a technologie Java. Ukázkou práce s tímto rozhraním je obr. 9.2:




*Obrázek č. 9.2: Grafický aplet v MapleNetu*

## 9.2 Maple T.A.

Systém Maple T.A. je určen pro procvičování, zkoušení a hodnocení studentů. Jde o webově orientovaný informační systém, jehož veškerá správa je prováděna přes Internet. Systém nevyžaduje ke své instalaci program Maple. K práci se systémem není zapotřebí žádné konfigurace.

### 9.2.1 Práce učitele v systému Maple T.A.

Pro každou tematiku je v rámci Maple T.A. administrátorem vytvořen samostatný účet, který učitel daného předmětu dále využívá. Po vytvoření tohoto účtu může odpovědný učitel vytvořit systém zkoušení. Pro administraci daného předmětu se používá rozhraní znázorněné na obr. 9.3:

*Obrázek č. 9.3: Administrace předmětu*

Toto rozhraní obsahuje čtyři důležité části:

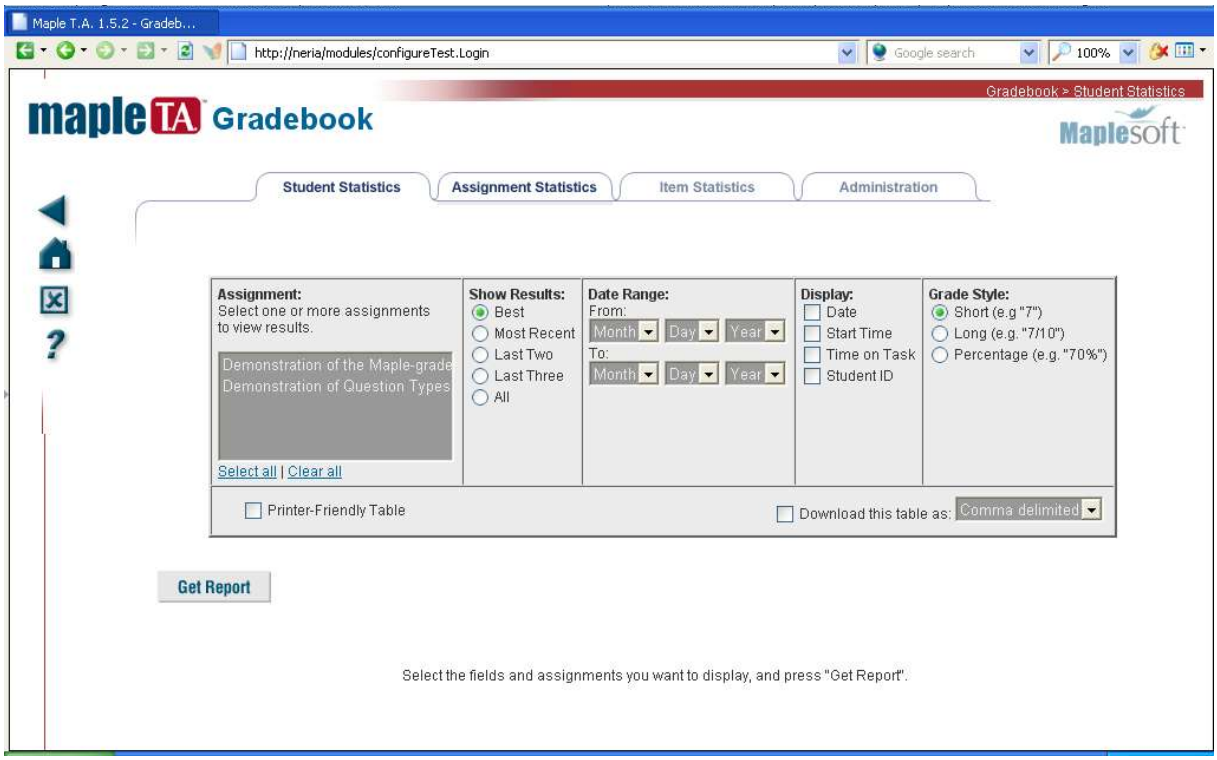
- *Tvorba databáze otázek* (Question Bank Editor)
- *Rozvržení otázek do cvičení a úkolů* (Assignment Editor)
- *Hodnocení výsledků* (Gradebook)
- *Systémové nástroje* (Systém Tools)

Tvorba systému zkoušení probíhá ve dvou fázích. V první fázi je vytvořena databáze otázek a odpovědí, ve druhé části jsou otázky rozvrženy do jednotlivých cvičení, domácích úkolů a testů.

Při vytváření otázek je možné vybrat z mnoha možností formy otázky. Mezi základní patří výběr z jedné nebo více možností, výběr z různých grafů a pochopitelně zadání odpovědi ve formátu Maple. Tato možnost je specifická zvláště tím, že vyžaduje po studentech nalezení vlastního řešení, přičemž není důležité, v jaké formě student řešení vytvoří, pomocí nástrojů Maple je lze porovnat s originálním řešením a určit jeho pravdivost.

Druhou fází je rozvržení otázek do jednotlivých cvičení, domácích úloh a testů. Tento proces je vícestupňový a umožňuje zařadit pro každý příklad více otázek, které jsou při zkoušení náhodně vygenerovány. Pochopitelně je možné zařadit jednu otázku do více cvičení, úloh nebo testů a tímto způsobem motivovat studenty k procvičování nehodnocených cvičení.

Maple T.A. obsahuje také několik nástrojů pro hodnocení a kontrolování výsledků studentských prací, pro analýzu statistik a pro administraci jednotlivých studentů. Celý systém hodnocení je rozčleněn do čtyř základních skupin – viz obr. 9.3:




*Obrázek č. 9.3: Nástroj pro přehled hodnocení*

**I. Hodnocení studentů:** V této skupině si může učitel pomocí přehledné tabulky prohlížíí výsledky jednotlivých studentů podle vybraných domácích úloh a testů. V tabulce jsou uvedeny nejen získané bodové hodnoty, ale k jednotlivým úkolům a testům je možné podle významu přiřadit procentuální hodnotu a tak jednoduše zjistit celkové výsledky.

**II. Statistický přehled výsledků:** Tabulka zobrazená v rámci této skupiny obsahuje statistickou analýzu nejen hodnocených úkolů a testů, ale i nehodnocených cvičení. V přehledu jsou uvedeny například maximální a minimální výsledky, aritmetický průměr nebo medián. Tyto hodnoty umožňují učiteli nalézt oblasti, ve kterých mají studenti problémy, a znovu je zopakovat.

**IV. Výsledky jednotlivých příkladů**: Zde jsou do zobrazeny podrobnosti o výsledcích jednotlivých příkladů v rámci vybraného úkolu nebo testu. Vzhledem k tomu, že příklady jsou vybrány z různých oblastí, je možné přesněji určit a nalézt specifické oblasti nutné ke zopakování.

**V. Administrace studentů**: V této části může učitel procházet výsledky jednotlivých studentů, zjišťovat stupeň jejich znalosti a podle výsledků studenty dále směřovat. Současně je možné vybranému studentovi nebo vybrané skupině studentů napsat e-mail. Specifikem je možnost upravit výsledky hodnocení jednotlivých příkladů s ohledem na přesnost odpovědi.

### 9.2.2 Práce studenta v systému Maple T.A.

Základem práce studenta v rámci systému Maple T.A. je jeho registrace do vybraného předmětu. Poté, co vyplní základní údaje pro systém se může přihlašovat do jednotlivých domácích úkolů a testů, přičemž systém si o výsledcích jednotlivých příkladů vede záznamy. Studenti, kteří si pouze chtějí své znalosti zopakovat a procvičit, aniž by potřebovali následné vyhodnocení, se nemusí v Maple T.A. zaregistrovat, protože procvičení nevyžadují přihlášení.

Při zpracování domácích úkolů a testů postupuje student podle pořadí jednotlivých příkladů, přičemž se k nim může vracet. V rámci jednoho příkladu může být více variant otázek, navíc jsou odpovědi na jednotlivé otázky promíchány.

Maple T.A. automaticky vyhodnocuje znalost studentů z probrané matematické látky, přičemž umožňuje přejít od jednoduchých otázek zadaných výběrem po náročnější otázky vyžadující vlastní řešení studenta ve formě výsledného výrazu. V takovém případě je systém Maple T.A. schopen zjistit ekvivalenci mezi originálním řešením a řešením, které zadal student.

Praktické problémy prezentované a hodnocené pomocí Maple T.A. prohlubují znalosti studentů a testují čemu studenti rozumí a co nepochopili. Automatické hodnocení cvičení a testů umožňuje zaznamenávat pokroky ve znalostech studentů a umožňují rovněž učitelům odpovídajícím způsobem změnit své učící techniky.

# Kapitola 10

## Práce s nápovědou a slovníkem

V této kapitole se budeme věnovat oblastem, které umožňují zjednodušit a zpřehlednit práci se systémem Maple 9.5 – hypertextovou nápovědou provázanou s matematicko-technickým slovníkem. Systém nápovědy je v programu Maple 9.5 velmi rozsáhlý a hluboce strukturovaný, a proto mu věnujeme samostatný oddíl. Jako novinka se v systému Maple 9.5 objevil slovník matematických a technických výrazů, který zajišťuje lepší pochopení jednotlivých funkcí.

## 10.1 Struktura nápovědy

Celá nápověda je členěna do několika základních oblastí podle způsobu práce a hledání, přičemž až na položku *History* odpovídají jednotlivým položkám v menu Help hlavního okna:

- ***Table of contents*** / *Obsah* (klávesová zkratka Ctrl-F1) – tato část obsahuje seznam témat nápovědy přehledně uspořádaných a členěných do formátu stromu.
- ***Topic search*** / *Vyhledávání témat* (klávesová zkratka Ctrl-F2) – zde je možné vyhledávat v seznamu jednotlivých témat nápovědy podle jejich názvů.
- ***Search*** / *Vyhledávání* (klávesová zkratka Ctrl-F3) – jde o standardní full-textový vyhledávač, který umožňuje prohledávat obsah textů nápovědy a také v matematické slovníku.
- ***Math Dictionary*** / *Matematický slovník* (klávesová zkratka Ctrl-F11)– obsahuje nový, plně integrovaný slovník matematických a technických výrazů.
- ***History*** / *Historie témat* – umožňuje procházet seznamem dosud prohlížených položek.

Tyto části nápovědy jsou vzájemně hypertextově provázány, což umožňuje rychlé a přehledné hledání, takže při vyhledávání klíčových slov se dozvíme nejen v jaké funkci jsou použity, ale také jaký význam má daný matematický význam.

## 10.2 Ovládání nápovědy

Okno nápovědy je možné otevřít mnoha způsoby – lze je otevřít jako dotaz na syntaxi známého příkazu, jako hledání v určité oblasti nebo při hledání významu některého matematického pojmu.

Okno nápovědy obsahuje jednoduché menu, které ve svých položkách obsahuje stejné informace jako standardní menu, pouze menu „File“ obsahuje položky umožňující tisku obsahu témat nebo celé nápovědy.

Další řádek obsahuje tlačítka pro zjednodušení ovládání – tisk témat, kopírování obsahu daného témata nápovědy, posun v historii nápovědy a skok na základní témata.

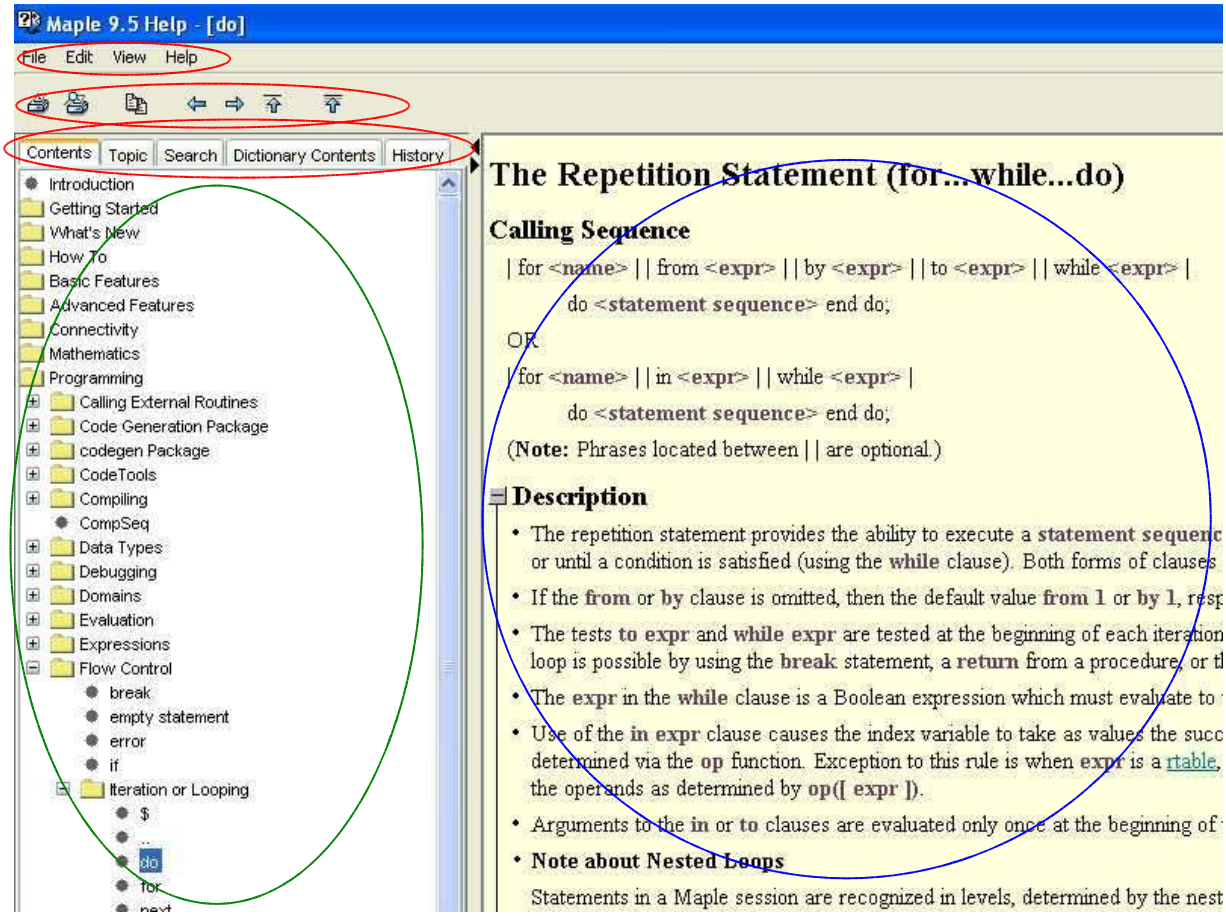
Nejdůležitější částí celé nápovědy jsou dvě okna v hlavní části. Vzájemnou velikost těchto dvou oken je možné měnit posunem lišty, která je odděluje, čímž můžeme zajistit větší prostor pro text nápovědy.

Levé okno obsahuje seznam témat, ze kterých je možné vybírat, přičemž je rozděleno záložkami podle způsobu práce. Tyto záložky odpovídají výše uvedeným částem struktury (*Contents, Topic, Search, Dictionary, History*). Téma, které nás zajímá, vybereme myší. V některých případech jde o celou oblast, kterou můžeme rozbalit a v této oblasti vybírat.

Pravé okno pak obsahuje vlastní text nápovědy je rozdělený do několika částí, které umožňují snazší orientaci, přičemž některé části nemusí být vždy zobrazeny:

- ***Calling sequence*** – jakým způsobem a s kolika parametry je daná funkce volána.
- ***Parameters*** – jakého typu mají být vstupní a výstupní parametry funkce.
- ***Description*** – prováděcí skupina s podrobným popisem funkce, tj. základní chování funkce, typ a formát výsledku, způsob vyhodnocování parametrů, způsob použití parametrů, volitelné parametry, omezení funkce a další.
- ***Examples*** – tato prováděcí skupina obsahuje ukázkové příklady použití funkce v různých situacích a s různými vstupy.
- ***See also*** – obsahuje seznam funkcí, které jsou dané blízké dané oblasti, nebo danou oblast zahrnují a případně jsou v ní zahrnuty.

Na obrázku 10.1 je zobrazen vzhled nápovědy v systému Maple 9.5 pro dotaz na příkaz *for*. Červenou barvou jsou znázorněny řídící prvky – dva řádky menu a lišta se záložkami. Zelenou barvou je znázorněno okno se seznamem témat pro výběr, zatímco pravé okno obsahuje text nápovědy. Jak je zřejmé, není zobrazena část *Parameters*. Mezi levou a pravou stranou je lišta pro změnu rozměru.




*Obrázek 10.1: Vzhled nápovědy pro příkaz* for

V následujících oddílech se budeme zabývat ovládáním jednotlivých oblastí a následnou prací s obsahem nápovědy.

### 10.2.1 Obsah témat (Table of Contents)

Celá nápověda je v systému Maple rozdělena do čtrnácti základních oddílů, které se týkají nejen jednotlivých příkazů, ale také práce a ovládání celého systému. Obsah také obsahuje dva přímé odkazy – *Introduction* a *Example Worksheets*. Uvádíme přehled těch nejdůležitějších:

- ***Getting Started*** – úvodní informace a rady pro nové uživatele systému Maple.
- ***What's New*** – informace pro ty uživatele, kteří již se systémem Maple pracovali a potřebují se informovat o posledních změnách v systému.
- ***How To*** – základní postupy při tvorbě programů a způsob práce.
- ***Basic Features*** – přehled ovládání aplikace Maple a práce s jeho editorem.
- ***Mathematics*** – seznam všech matematických funkcí rozdělený do hlubší struktury pododdílů, které identifikují jednotlivé oblasti matematiky, jako je analýza, diferenciální počet, algebra, logika, statistika, finanční funkce a další. Zde jsou informace například o funkcích používaných k řešení diferenciálních rovnic.
- ***Programming*** – programátorská příručka. Obsahuje důležité informace týkající se používání proměnných, volání funkcí z balíků, definici vlastních funkcí a balíků, řízení programu a mnoha dalších problémů. Příkladem je výše uvedená nápověda k řídícímu příkazu cyklu *for*.
- ***Graphics*** – seznam těch funkcí, které vytváří grafický výstup, jako je například funkce *plot.*

### 10.2.2 Vyhledávání témat (Topic search)

Při výběrů této záložky je možné vyhledávat v jednotlivých tématech nápovědy. Tímto způsobem je možné hledat pouze v názvech jednotlivých položek obsahu, což je výhodné v situacích, kdy neznáme přesné jméno hledané funkce, pouze jeho část.

Vyhledávání v tématech je také vhodné využít tehdy, pokud hledáme všechny funkce, které řeší náš problém, známe základní tvar anglického názvu, a současně nechceme být zahlceni množstvím informací nalezených pomocí běžného fulltextového vyhledávání.

### 10.2.3 Vyhledávání (Search)

Při použití tohoto nástroje se provádí vyhledávání ve všech plných textech obsažených v nápovědě. Tímto způsobem je možné nalézt všechny záznamy týkající se dané problematiky, současně je však možné nalézt i ty záznamy, které se daného problému týkají pouze částečně.

Je dobré dávat pozor na to, že vyhledávání je v Maple implementováno jako *or-search*, což znamená, že jsou vyhledány všechny záznamy obsahující aspoň jedno ze zadaných slov.

### 10.2.4 Historie témat (History)

Obsahuje seznam všech témat, která uživatel otevřel. Tímto způsobem se lze rychle vrátit nebo rychle posunout dopředu na ta témata, která jsou aktuální. Toho lze využít například v situacích, kdy současně používáme několik funkcí a potřebujeme znát jejich přesnou syntaxi.

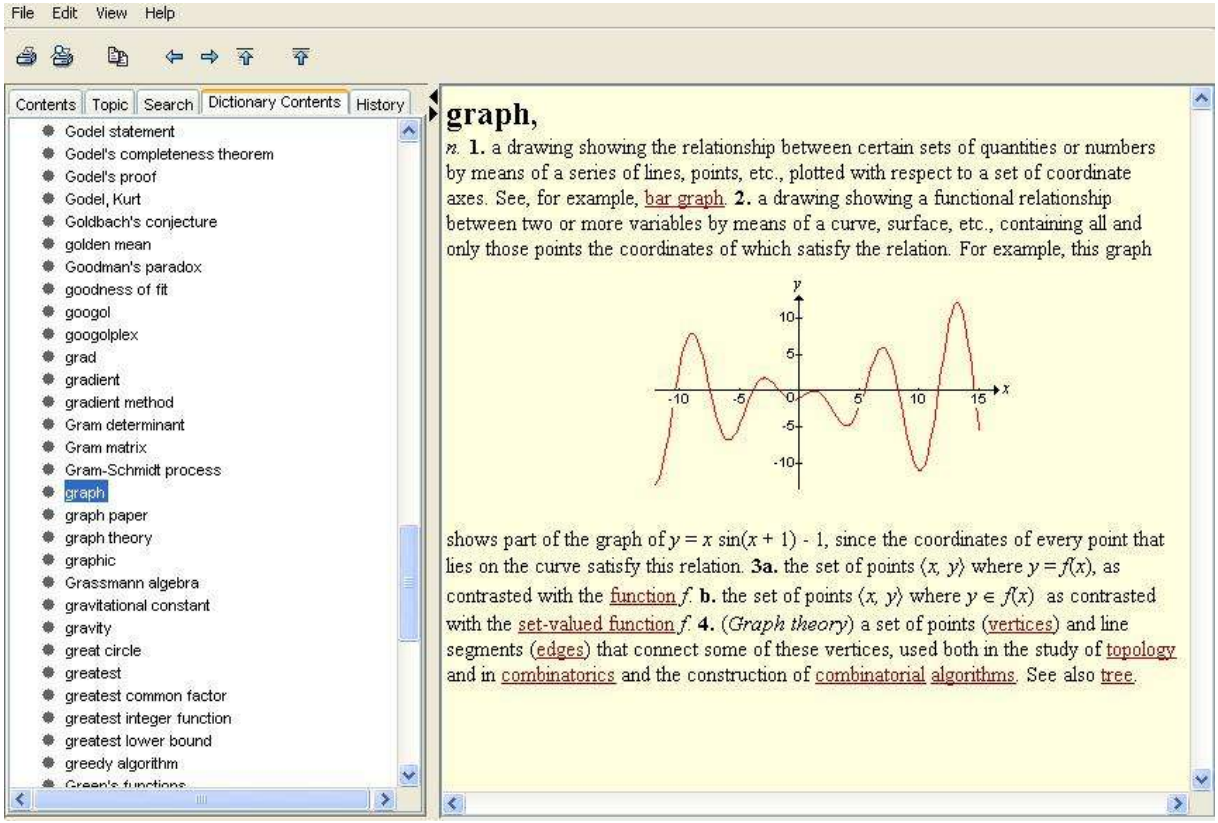
## 10.3 Matematický slovník (Math Dictionary)

Maple 9.5 obsahuje jako novinku velký slovník matematických a technických výrazů. Tento slovník obsahuje přibližně pět tisíc pojmů, které jsou provázány hypertextovými odkazy. Navíc obsahuje přibližně třista diagramů, obrázku a matematických vzorců.

Pojmy matematického slovníku je možné nalézt několika různými způsoby:

- Přímým procházením seznamu pojmů. Pojmy jsou řazeny do složek podle prvního písmena, takže v nich lze rychle vyhledávat.
- Použitím hypertextového odkazu z nápovědy. Všechna témata nápovědy obsahují – pokud je daný pojem obsažen v textu nápovědy – odkazy na pojmy definované ve slovníku. Tyto odkazy jsou barevně zvýrazněny, a navíc je možné si zobrazit stručnou verzi najetím myši na daný odkaz.
- Použitím příkazu „?“ na příkazové řádce, podobně jako pro libovolný jiný výraz.
- Využitím vyhledávání v tématech nápovědy.

Vzhled matematického slovníku pro pojem *graph* je zobrazen na obrázku 10.2:




*Obrázek 10.2: Matematický slovník*

# Literatura

Buchar J., Hřebíček J., Hřebíčková J., Slaběnáková J. 1994, *Úvod do programového souboru MAPLE V*, Vysoká škola zemědělská v Brně, Brno.

Cikalo S. 2004, Nové možnosti vědeckých výpočtů v Maple, Diplomová práce, Fakulta informatiky MU v Brně, Brno.

Dalík, J., Hřebíček, J., Hřebíčková, J., Ráček, J., Slaběňáková, J. 2001, Využití informační technologie Maple ve výuce matematiky a ekonometrie. In *Sborník Informační technologie pro praxi. Ostrava : Tanger, s.r.o*, s. 77-82.

Dalík J., Hřebíček J. 2001, Využití počítačového systému Maple ve výuce matematiky na Fakultě stavební VUT v Brně. In *Sborník 2. konf. o matematice a fyzice na vysokých školách technických, Vojenská akademie v Brně*, s. 30-35.

Dočkal, J., Doupovec, M. 2000, MATEMATIKA III – inovace s podporou systémů symbolické matematiky. Fond rozvoje. Projekt 1578, Ústav matematiky FS VUT Brno, Brno.

Dočkal, J. 2004, Numerické metody s Maple v kombinovaném studiu, In *Sborník 9. konference Pedagogický software 2004, České Budějovice*, s. 209-212.

Došlá, Z., Plch, R., Sojka, P. 2002, *Matematická analýza s programem Maple. Díl 2, Nekonečné řady*. Vyd. prvni. Brno : Masarykova Universita, 2002. 453 s.

Dufek J. & Hřebíček J. 2000, Paradigm of Solving Problems in Econometrics. *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis*. Vol. XLVIII, No. 2, p. 33-36.

Gander, W. & Hřebíček, J. 2004, *Solving Problems in Scientific Computing Using Maple and MATLAB*. 4th, expanded and rev. ed., 476 p. Springer, Heidelberg.

Hašek, R., Klufová, R., Rost, M., 2002, Analýza prostorových dat v programu Maple, In *Zborník vedeckých prác z medzinárodnej vedeckej konferencie Matematika vo výučbe, výskume a praxi,* SPU Nitra, s. 255 - 259.

Hřebíček J., Pitner T., Buchar J.1997, Computational Simulation Using Maple, In *Proceedings International Summer School Computer Aided Education in Automation and Control (editor M. Huba)*, Slovak University of Technology Bratislava, s. 98-106.

Hřebíček, J. & Ráček, J. 2001, *Matematika v obrazech.* Fakulta informatiky MU v Brně, 24 s. Brno, skripta pro distanční vzdělávání.

Hřebíček, J. & Ráček, J. 2001, *Symbolická matematika*. Fakulta informatiky MU v Brně, 38 s. skripta pro distanční vzdělávání.

Hřebíček, J. 2002, Využití Maple na vysokých školách v ČR. In *Sborník Využití Maple ve výuce a výzkumu na vysokých školách a akademiích věd. Brno, ECON Publishing*, s. 8, Brno.

Hřebíček, J., Pešl, J., Ráček, J.: Using Symbolic Computing Systems in Applications. In *Proceedings of International Conference The Decidable and the Undeciable in Mathematics Education. Brno: The Mathematics Education into the 21th Century Project 2003*, 19--25. September, 2003, Brno: VUT Brno, p. 116-120.

Hřebíček, J., Hřebíčková, J., Mezník, I., Chvátalová, Z. 2004, E-learning in Teaching Mathematics Using Computer Algebra System Maple. In *Proceedings of the International Conference The Future of Mathematics Education, ed. A. Rogerson*. Ciechocinek, Poland.

Hřebíček, J., Kohout, J., Mezník, I., Chvátalová, Z. 2004, Využití e-lerningových nástrojů Maple při výuce matematického modelování. In *Zborník 28. konferencie VŠTEP*, JSMF, Slovenská matematická společnost, Žilinská univerzita EDIS s. 107-118.

Hřebíček, J., Kohout., J., Hřebíčková, J., Mezník, I., Chvátalová, Z. 2004, Elektronická podpora výuky matematiky s využitím systému počítačové algebry Maple. In *Sborník 9. konference Pedagogický software 2004, České Budějovice*, s.205-208.

Klufová, R., Hašek, R., 2002, Využití programu Maple V při výuce matematických základů geoinformatiky, In *Sborník přednášek a programů z 9. ročníku mezinárodní konference Pedagogický software 2002*, JČU Č. Budějovice, s. 6.
Koudelák V. 2003, Přehled zdrojů informací na webu o systémech počítačové algebry, Bakalářská práce, Fakulta informatiky MU v Brně, Brno
Němeček, A. 2004, Nové možnosti programu Maple ve výuce matematiky", In *Zborník 28. konferencie VŠTEP*, JSMF, Slovenská matematická společnost, Žilinská univerzita, p. 249-256.
Němeček, A. 2002, Numerické metody v přímém přenosu, In *Sborník Využití Maple ve výuce a výzkumu na vysokých školách a akademiích věd. Brno, ECON Publishing*, s. 16, Brno
Plch R. 1998, Diferenciální počet funkcí více proměnných s programem MapleV, disertační práce, Masarykova univerzita v Brně, Brno.
Plch, R., Došlá, Z., Sojka, P. 1999*, Matematická analýza s programem Maple. Díl 1, Diferenciální počet funkcí více proměnných*. Vyd. prvni. Brno. 8 s. CD-ROM.
Plch, R. 2004, Matematická analýza s programem Maple. In *3rd International Conference Aplimat*. Bratislava : Department of Mechanical Engineering, Slovak University of Technology in Bratislava, s. 789-792.
Rosický J. 2002, Nové možnosti výpočtu v Maple pro studenty, Bakalářská práce, Fakulta informatiky MU v Brně, Brno
Soldánová E. 2003, Nové možnosti Maple 8 ve výuce matematiky, Bakalářská práce, Fakulta informatiky MU v Brně, Brno
Stryk L. 2001, Číselné obory v systému Maple, Bakalářská práce, Fakulta informatiky MU v Brně, Brno

http://www.maplesoft.com/ - stránky Waterloo Maple Inc.