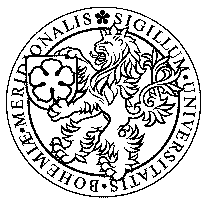
Pedagogická fakulta Jihočeské univerzity
České Budějovice

katedra matematiky



Využití programu Maple při výuce analytické geometrie

diplomová práce

Vypracovala: Petra Kubišová
Vedoucí dipl. práce: Mgr. Roman Hašek, Ph.D.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně s použitím uvedené literatury.

Petra Kubišová

……

Ve Strakonicích 20. 4. 2007

Děkuji vedoucímu mé diplomové práce Mgr. Romanu Haškovi za skvělý nápad použít Maple a v něm vytvořit aplikace – maplety o analytické geometrii, inspiraci a za odborné vedení.

Anotace

Název: Využití programu Maple při výuce analytické geometrie
Vypracovala: Petra Kubišová
Vedoucí práce: Mgr. Roman Hašek, Ph.D.

Klíčová slova: Analytická geometrie, výuka pomocí počítačů, programování, aplety, Maple.

Obsahem této práce je vytvoření aplikací nazvaných maplety pro ukázání vlivu jednotlivých koeficientů z rovnic analytické geometrie na jejich zobrazení. Maplety jsou naprogramovány v programu Maple. Tyto pomůcky by měly usnadnit výuku analytické geometrie na gymnáziích a v úvodních kurzech vysokoškolské geometrie.

Title: Applying of Maple in analytical geometry
Author: Petra Kubišová
Supervisor: Mgr. Roman Hašek, Ph.D.

Key words: Analytical geometry, teaching with software, programme, aplication, Maple.

Content of this diploma is create aplications called maplets. Maplets are for showing influences each coefficients from equation to their representation. Maplets are programmed in the program Maple. These tools should make easy teaching analytical geometry in high schools and in basic courses at university.

OBSAH

1. Úvod

Volba tématu

Má diplomová práce nese název Využití Maple při výuce analytické geometrie. Zaměřila jsem se zvláště na tvorbu apletů v programu Maple a na vizualizaci rovnic analytické geometrie. Vedly mě k tomu následující důvody:

- počáteční problémy studentů s představou významu parametrů v rovnicích útvarů
- výuka analytické geometrie při náslechové praxi na gymnáziu S. Jirsíka v Č. Budějovicích
- účast na matematické konferenci Užití počítačí ve výuce matematiky konané v Č. Budějovicích ve dnech 10.-12.11.2005, hlavně příspěvky o programu Maple a o mapletech včetně wokrshopu o tomto programu.

Cíle

Cílem této práce je prozkoumání využití programu Maple 9.5 ve výuce analytické geometrie nejen na gymnáziích, ale i v úvodních kurzech vysokoškolské geometrie, vytvoření demonstračních pomůcek v tomto programu, které by usnadnily výuku analytické geometrie. Vzhledem k širokému uplatnění Maple v celé matematice jsem se více soustředila na prozkoumání knihoven a příkazů týkajících se počítání a zobrazování v analytické geometrii. Poté jsem se soustředila na programování samotných mapletů, případně spojení zobrazování a počítání v analytické geometrii s tvorbou mapletů. Sledovala jsem především vizualizaci rovnic analytické geometrie.

Soustředila jsem se na dva dílčí cíle:

- Vytvořit balíček demonstračních pomůcek pro výuku analytické geometrie, s nimiž by mohli pracovat sami studenti.

- Navrhnout a popsat, jak by si tyto a podobné pomůcky mohl každý zájemce co nejsnadněji vytvořit sám.

O obsahu

Nejdříve shrnu náplň středoškolské analytické geometrie a navazujících úvodních kurzů na vysokých školách. Poté vás seznámím s programem Maple a jeho základním ovládáním. Tento stručný popis by měl nezasvěceným čtenářům v hrubých rysech přiblížit použitý program a tím i mou práci. Dále se zaměřím je na tvorbu mapletů. V páté kapitole uvádím nejdříve přehled všech vytvořených pomůcek a pak je stručně popíšu. Následují návrhy na využití těchto pomůcek. Použité zkratky a matematické symboly jsou vysvětleny na konci této práce.

Nedílnou součástí mé práce je samozřejmě kompaktní disk obsahující balíček souborů, které jsem vytvořila jako pomůcku pro výuku analytické geometrie.

2. Metodika

S obsahem středoškolské analytické geometrie jsem se seznámila prostřednictvím osnov matematiky pro gymnázia, několika středoškolských učebnic matematiky obsahujících kapitolu analytická geometrie. S obsahem úvodních kurzů vysokoškolské analytické geometrie jsem se seznámila prostřednictvím několika přednášek a skript k danému tématu. Prostudovala jsem samozřejmě i didaktickou literaturu ([5], [6], [9]).

Moje práce s programem Maple se dá shrnout do několika etap:

- seznamování s programem, zvládnutí ovládání programu
- studium manuálů k programu, zdrojových kódů z nápověd programu, zvládnutí programování apletu v Maple
- hledání možností využití programu ve výuce, výběr témat analytické geometrie ke zpracování v Maple
- vytvoření pomůcek v Maple pro výuku zvolených témat
- sestavení návodů pro vytvoření mapletů v programu

Postupovala jsem především formou samostudia. Základy práce s programem Maple jsem získala v kurzu Výpočetní technika pro matematiky a hlavně v kurzu Řešení problému užitím programů Derive a Maple.

Výběr zpracovaných témat je především pro zlepšení představ významu jednotlivých koeficientů v analytických vyjádřeních různých útvarů. Vizualizace rovnic přímek, kuželoseček, rovin a ploch přispěje k zlepšení prostorové představivosti studentů.

3. Analytická geometrie

3.1 Výuka analytické geometrie na gymnáziích

Dle [5], [6], [9] patří do gymnaziální analytické geometrie toto základní učivo:

1. základní pojmy analytické geometrie v rovině

- soustava souřadnic v rovině
- vzdálenost bodů, střed úsečky
- orientovaná úsečka, vektor, souřadnice vektoru, velikost vektoru
- sčítání vektorů a násobení vektoru reálným číslem, lineární závislost a nezávislost vektorů, skalární součin vektorů
- parametrické vyjádření přímky, obecná rovnice přímky, směrnicový tvar přímky
- vzájemná poloha přímek, odchylka přímek, vzdálenost bodu od přímky
- analytické vyjádření kružnice, vzájemná poloha přímky a kružnice, tečna
- elipsa, parabola, hyperbola, jejich základní vlastnosti, konstrukce
- vrcholová rovnice paraboly, osová rovnice elipsy a hyperboly
- určení kuželosečky z jejího analytického vyjádření
- vzájemná poloha přímky a kuželosečky, tečny

2. analytická geometrie v prostoru

- soustava souřadnic v prostoru, souřadnice bodu a vektoru
- vzdálenost bodů, velikost vektoru
- operace s vektory v prostoru, lineární kombinace vektorů, vektorový součin
- parametrické vyjádření přímky a roviny v prostoru, obecná rovnice roviny
- vzájemná poloha bodů, přímky a roviny v prostoru
- vzdálenosti a odchylky
- kulová plocha

Je vidět, že se učivo opírá o značné matematické základy ze základní školy popř. z jiných předmětů (fyzika). Předpokládá také dobrou prostorou představivost a zvládnutí středoškolské stereometrie.

Posloupnost a další obsahové a časové řazení probíraných celků je dle osnov v plné kompetenci učitele.

3.2 Úvodní kurz kuželoseček a ploch na vysoké škole

Kuželosečky zadané obecnou rovnicí nemusejí být pouze elipsa, hyperbola, parabola, ale třeba také imaginární elipsa, různoběžky, rovnoběžky, přímka, imaginární různoběžky či imaginární rovnoběžky.

Kvadriky v R^3 jsou plochy o obecné rovnici ve tvaru

$$a_{11}x^2 + a_{22}y^2 + a_{33}z^2 + a_{12}xy + a_{13}xz + a_{23}yz + a_{14}x + a_{24}y + a_{34}z + a_{44} = 0$$

kde a_{11}, a_{22}, a_{33}, a_{12}, a_{13}, a_{23}, a_{14}, a_{24}, a_{34}, a_{44} jsou reálné konstanty.

Tento tvar rovnice určující kvadriku K lze převést na tzv. kanonický tvar. To znamená zvolit takovou ortonormální soustavu souřadnic (O; $\vec{i} \cdot \vec{j} \cdot \vec{k}$), v níž rovnice kvadriky K bude mít tvar

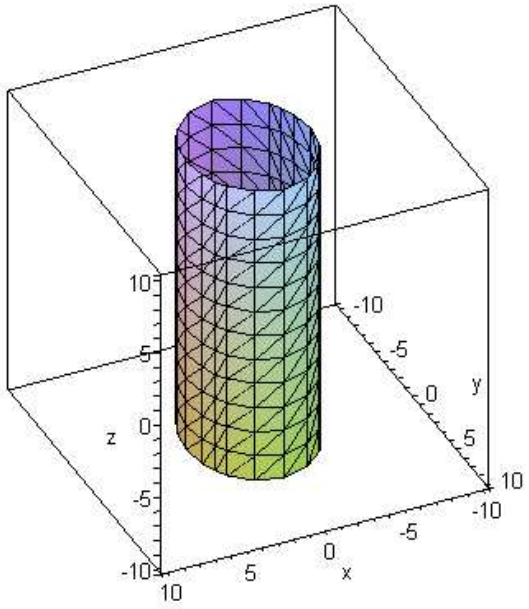
$$b_{11}x^2 + b_{22}y^2 + b_{33}z^2 + b_{14}x + b_{24}y + b_{34}z + b_{44} = 0$$

kde b_{11}, b_{22}, b_{33}, b_{14}, b_{24}, b_{34}, b_{44} jsou vhodné reální konstanty. Vidíme, že jsme takto odstranili smíšené členy xy, xz a yz. Takováto soustava souřadnic bude mít počátek O[0,0,0] jako vrchol kvadriky K a ortonormální vektory $\vec{i} \cdot \vec{j} \cdot \vec{k}$ budou určeny jednotkovými vektory hlavních směrů kvadriky K.

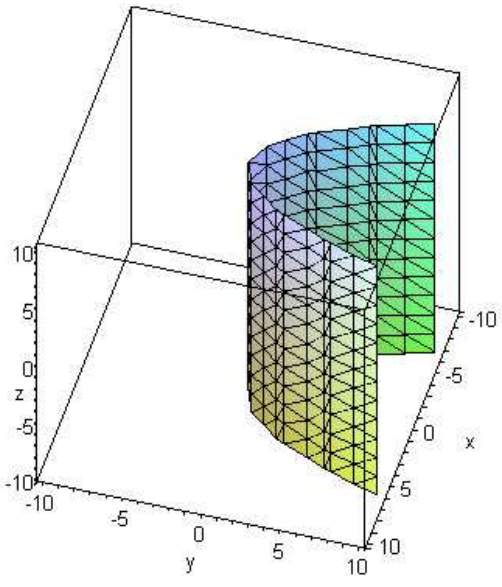
Jednotlivé kruhy kvadrik a jejich kanonické rovnice $a > 0, b > 0, c > 0, p > 0, q > 0, k \neq 0$.

- Válcové plochy, jejíž řídící křivka je kuželosečka

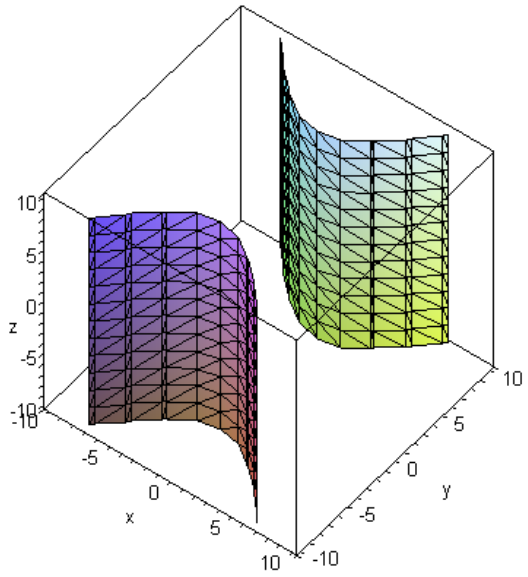
  Eliptický válec (eliptická válcová plocha) – řídící křivka je elipsa, rovnice je $\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$, kde a, b jsou reálná čísla.



  Parabolický válec (parabolická válcová plocha) – řídící křivka je parabola, rovnice je $y=\frac{x^2}{p}$, kde p je reálné číslo.

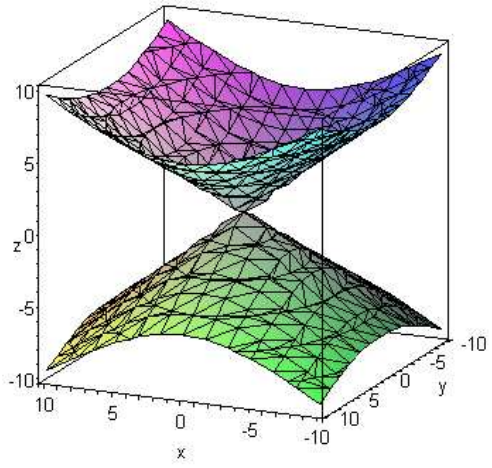


Hyperbolický válec (hyperbolická válcová plocha) – řídící křivka je hyperbola, rovnice je $-\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$, kde a, b jsou reálná čísla.

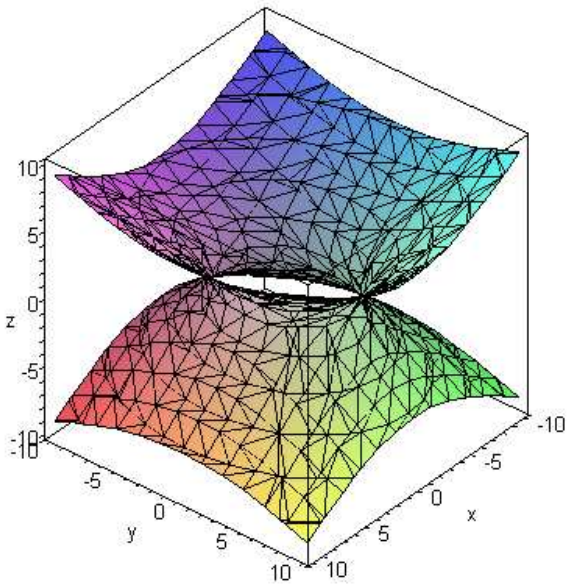


- Nerotační kvadriky vzniklé z rotačních vhodnou dilatací podél os

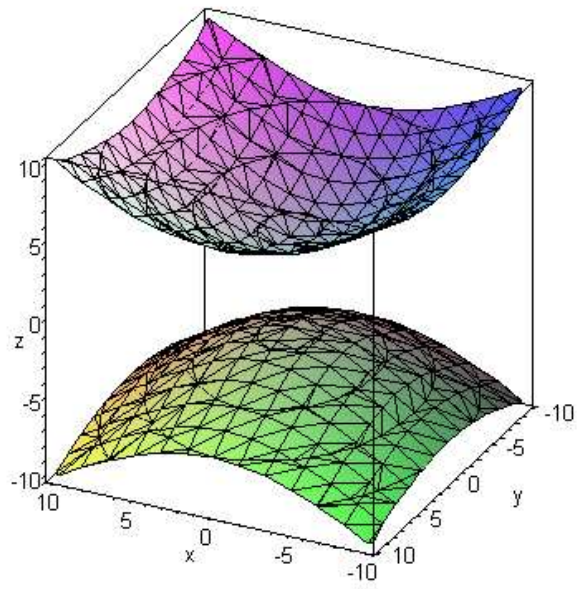
Eliptický kužel (eliptická kuželová plocha) – rotační kuželová plocha vzniká rotací různoběžných přímek kolem osy úhlu, který vytvářejí; rovnice je $\frac{x^2}{a^2}+\frac{y^2}{b^2}-\frac{z^2}{c^2}=0$, kde a, b, c jsou reálná čísla.



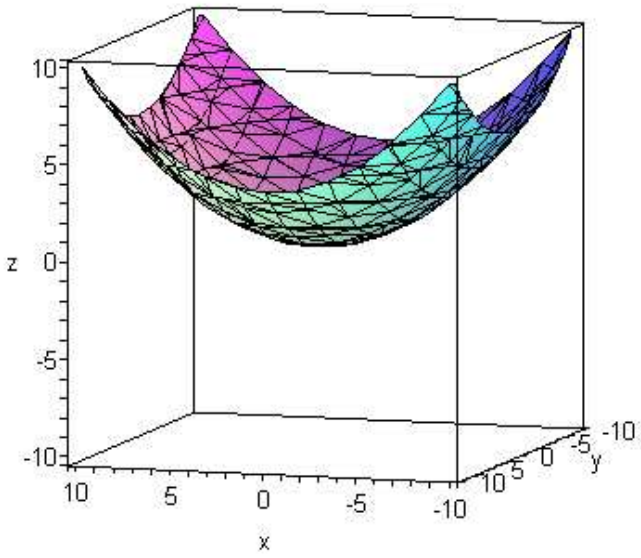
Jednodílný hyperboloid – rotační jednodílný hyperboloid vzniká rotací hyperboly kolem její vedlejší osy; rovnice je $\frac{x^2}{a^2}+\frac{y^2}{b^2}-\frac{z^2}{c^2}=1$, kde a, b, c jsou reálná čísla.



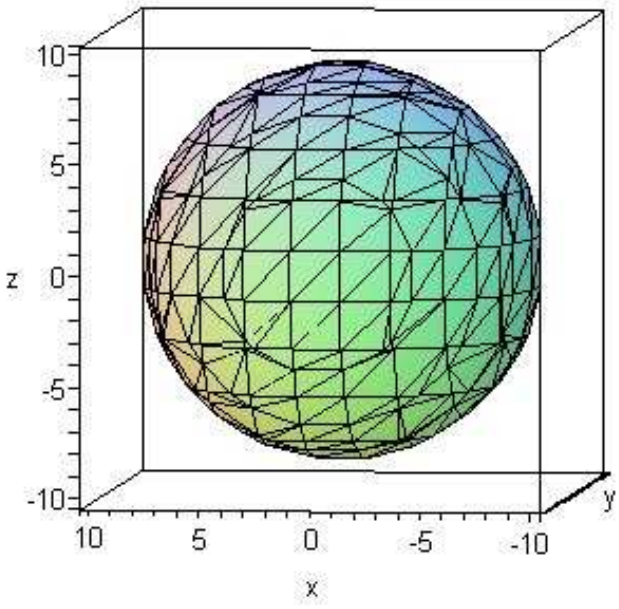
Dvoudílný hyperboloid – vzniká rotací hyperboly kolem její hlavní osy, rovnice je $\frac{x^2}{a^2}+\frac{y^2}{b^2}-\frac{z^2}{c^2}=-1$, kde a, b, c jsou reálná čísla.



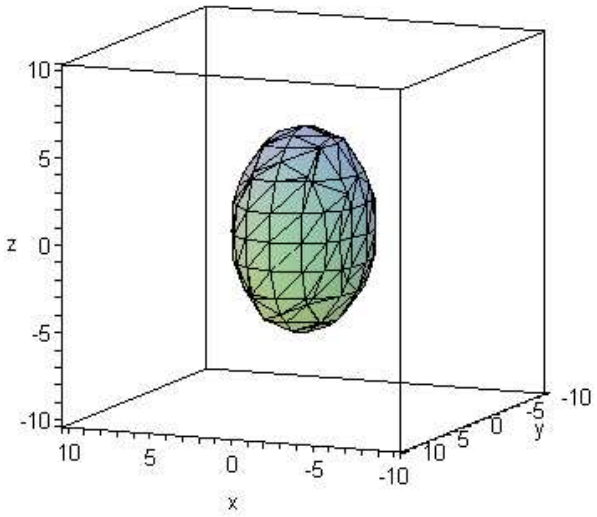
Eliptický paraboloid – vzniká rotací paraboly kolem její osy, rovnice je $z = \frac{x^2}{p} + \frac{y^2}{q}$, kde p, q jsou reálná čísla.



Kulová plocha – kulová plocha vzniká rotací kružnice kolem jejího průměru, rovnice je $x^2 + y^2 + z^2 = r^2$, kde r je reálné číslo udávající poloměr kulové plochy.

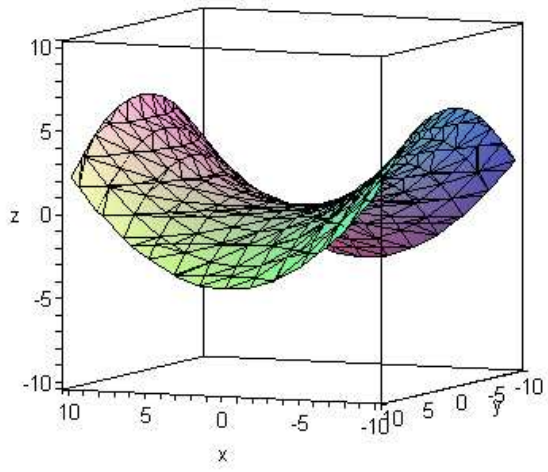


Elipsoid – vzniká rotací elipsy kolem jedné z jejích os, rovnice je $\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1$, kde a, b, c jsou reálná čísla.

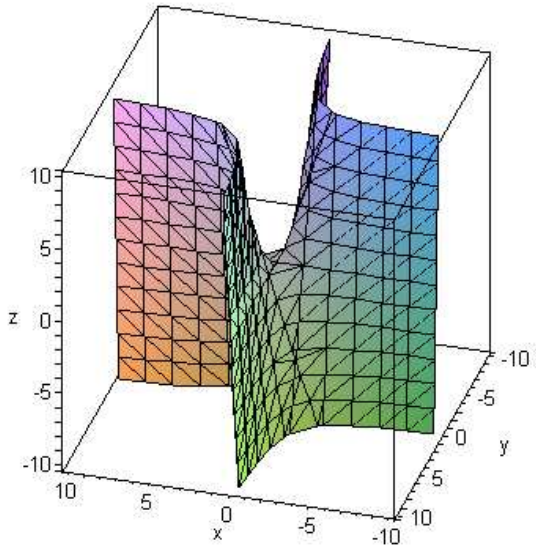


- Ostatní kvadriky

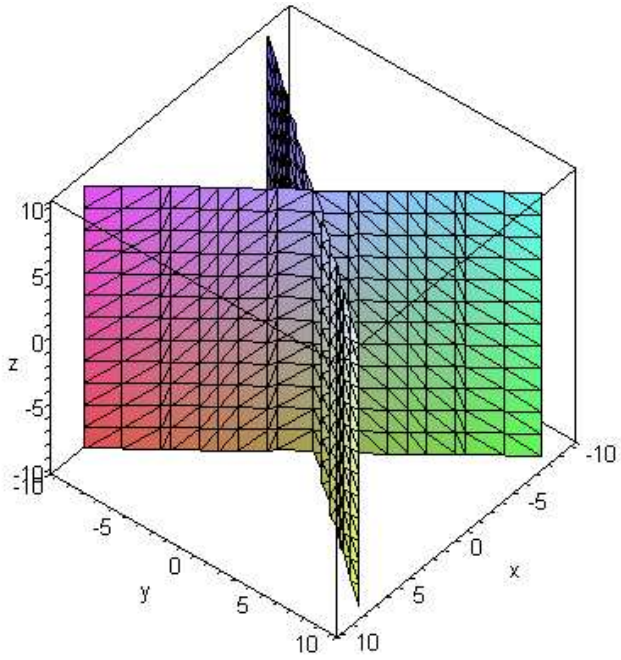
Hyperbolický paraboloid – pro představu posunujeme po parabole jinou parabolu $z=\frac{x^2}{p}-\frac{y^2}{q}$, kde p, q jsou reálná čísla.



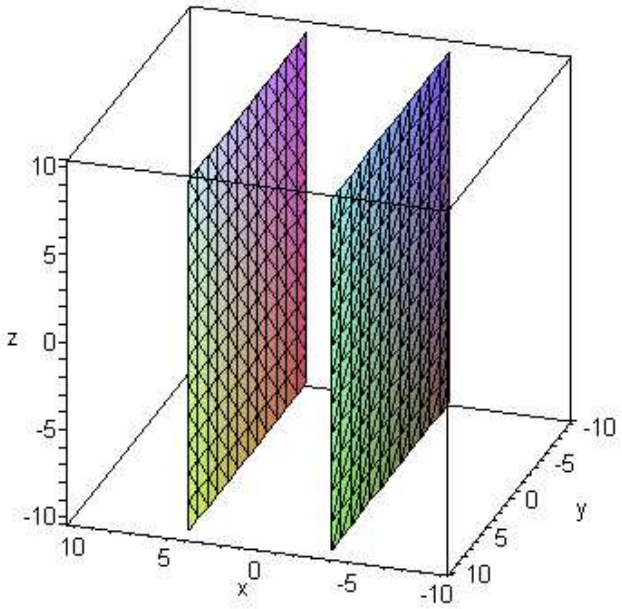
$z = kxy$, kde k je reálné číslo.



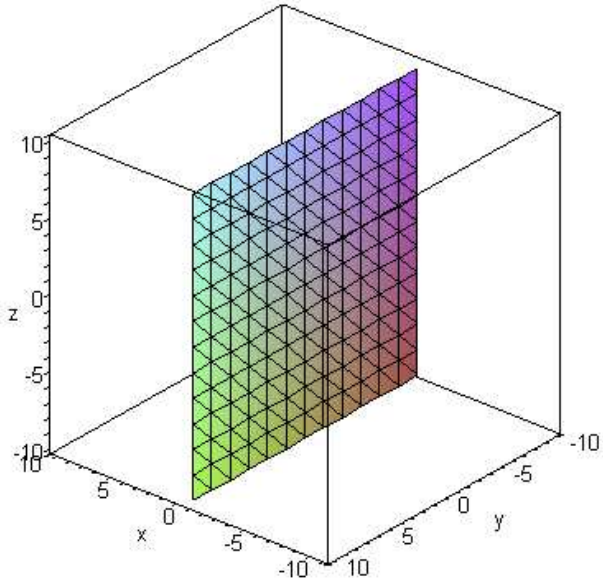
Dvě roviny různoběžné $\frac{x^2}{a^2} - \frac{y^2}{b^2} = 0$, kde a, b jsou reálná čísla.



Dvě roviny rovnoběžné $\frac{x^2}{a^2}=1$, kde a je reálné číslo.



Dvě roviny splývající $x^2=0$



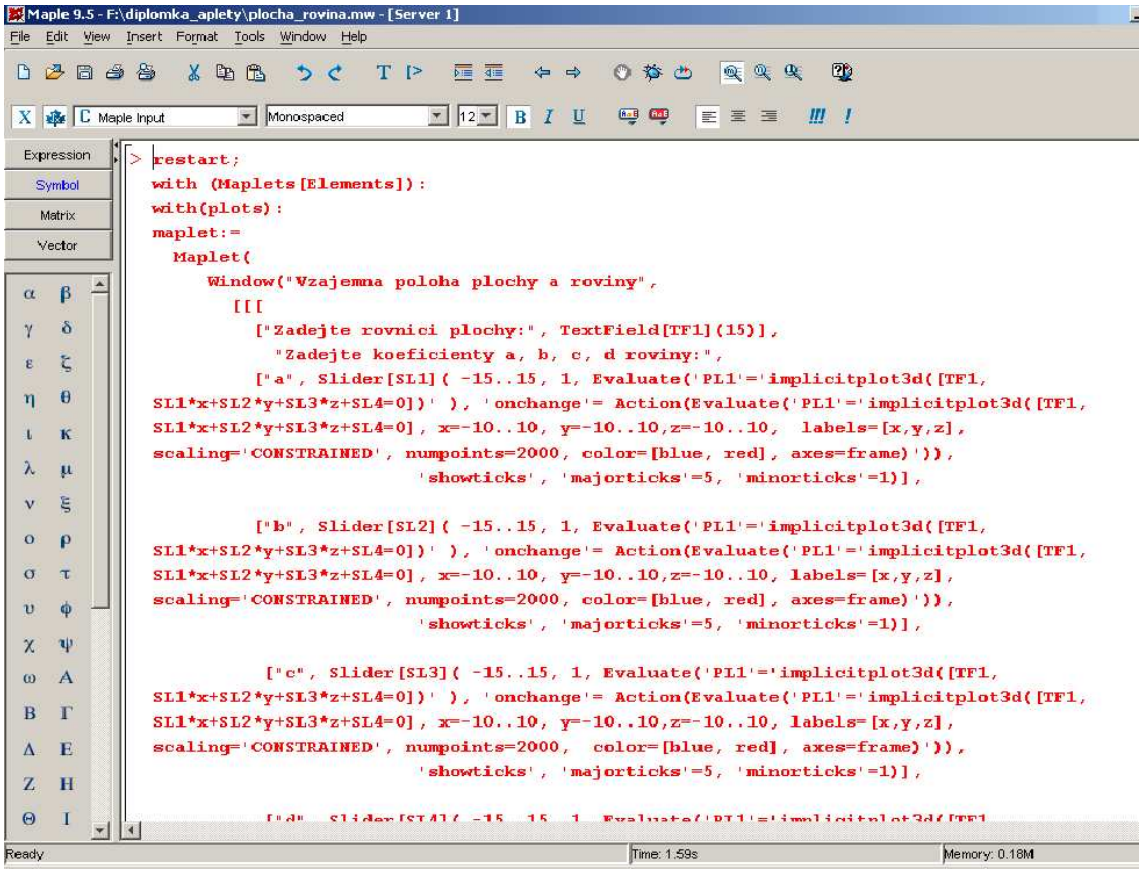
4. Maple

4.1. O programu

Program Maple je matematický program od firmy MapleSoft, který umožňuje pomocí příkazů řešit úlohy aritmetické, algebraické, grafické, statistické, úlohy kalkulu, diferenciálních rovnic, lineární algebry, geometrie, ale i grafické dvourozměrné i třírozměrné. Programem lze i programovat procedury, aplety a jiné. Aktuální verzi a cenu můžete najít na stránkách http://www.maplesoft.com. Program je v angličtině a obsahuje 2 grafické prostředí, jedno Maple 9.5 (s mnoha grafickými prvky navíc) a druhé Classic Worksheet (klasické z předcházející verzí). Já jsem pracovala v programu Maple 9.5.

4.2. Ovládání programu

Spuštěné okno programu Maple 9.5 vypadá následovně:



V horním řádku je zobrazena nabídka programu. Zleva je v nabídce: Soubor, Úpravy, Zobrazení, Vložit, Formát, Nástroje, Okno a Nápověda. Pod touto nabídkou je zobrazen panel nástrojů pro Nový soubor, Uložení, Tisk, Vyjmutí, Kopírování, Vložení, Úprava předchozí, Úprava následující, Text, Vložení řádku, Odsazení, Předsazení a další. Pod tímto panelem nástrojů je další panel nástrojů pro úpravu textu. S ním můžeme měnit font písma, velikost písma, zarovnání a další. V levé části obrazovky se nachází pomocné nástroje pro psaní příkazů do programu. Od shora jsou zde nástroje pro tvorbu Výrazů, Symboly, nástroje pro vytvoření Matic a Vektorů, jednotlivá písmena řecké abecedy. Největší část okna tvoří bílá plocha pro psaní příkazů. Dole na spodním řádku v okně programu je stavový řádek pro zobrazování informací o stavu programu.

S program komunikujeme pomocí příkazů. Ty se píší většinou všemi malými písmeny do řádku a ukončují se středníkem, popřípadě dvojtečkou. Tím se příkaz vykoná. Rozdíl je ale v tom, zda chceme, aby se výsledek zobrazil či nezobrazil. Pokud příkaz ukončíme středníkem, stiskneme Enter, příkaz se provede a výsledek vypíše na obrazovku, kurzor přeskočí na další prázdný řádek a čeká na další příkaz. Pokud příkaz ukončíme dvojtečkou, příkaz se provede, na obrazovce se nic neobjeví a kurzor přeskočí na další prázdný řádek a čeká na další příkaz.

Všechny zadané proměnné si program pamatuje. Chceme-li vymazat jejich hodnoty, musíme nový výpočet inicializovat příkazem *restart;*, který vymaže všechny proměnné. Vyhneme se tak problémům při opakování výpočtu.

Nápovědu si lze zobrazit několika způsoby. Například zadáním příkazu ve tvaru *?plot* – získáme nápovědu k příkazu *plot* i s odkazy na příbuzná témata. Jiný způsob je použít nápovědu programu (Help). Tu si zobrazíme z nabídkové lišty spuštěním Help – Introduction a dále podle hledaného výrazu, případně zapsáním klíčového slova do vyhledávání podle rejstříku. Nebo je možné umístit kurzor na příkaz, o kterém chceme podrobnosti, a

stisknout funkční klávesu F1, která spustí nápovědu programu a přímo nastavenou stránkou na náš hledaný příkaz.

Výpočty

Mezi nejjednodušší příkazy programu patří základní počítání s čísly. Může provádět sčítání, odčítání, násobení, dělení, umocňování, odmocňování a další operace s čísly či výrazy. Pro příklad

```
> 1+2*3^2;
```

$$19$$

```
> 5!;
```

$$120$$

```
> sqrt(2);
```

$$\sqrt{2}$$

```
> Pi;
```

$$\pi$$

```
> exp(1);
```

$$\mathbf{e}$$

Pro určení přibližné hodnoty použijeme příkaz *evalf()*, pro použití výsledku z předchozího výpočtu používáme znak %.

```
> evalf(sqrt(2));
```

$$1.414213562$$

```
> sqrt(7);
  evalf (%);
```

$$\sqrt{7}$$

$$2.645751311$$

Pro výpočet hodnoty obecně zadaného algebraického výrazu musíme znát konkrétní hodnotu dané proměnné ve výrazu. Byla-li proměnná již v programu použita, je k ní přiřazena první se vyskytující hodnota. Proto se zde často používá pro vymazání paměti s hodnotami proměnných příkaz restart. Výraz pro zadání konkrétní hodnoty pro konkrétní proměnnou pak vypadá následovně: název proměnné, znak pro přiřazení (:=) a hodnota proměnné.

```
> y:=5;
```

$$y := 5$$

Jednotlivé výrazy si můžeme pro další počítání označit. Poté můžeme používat jen zástupný znak či proměnnou. Program si při výpočtu sám dosadí dané číslo do pravého výrazu.

```
> f:=x->x^2;
```

$$f := x \to x^2$$

Pro výpočet stačí jen zapsat hodnotu x, pro kterou se daná funkční hodnota vypočítá. Pro náš příklad pak bude funkční hodnota rovna 4.

```
> f(2);
```

$$4$$

Pro řešení rovnice je základní příkaz *solve()*.

```
> solve(2*x+3=5);
```

$$1$$

Pokud nechceme řešení ve tvaru zlomků a odmocnin, ale v desetinném čísle, můžeme použít *evalf()*. Jednotlivé příkazy lze do sebe vnořovat.

```
> solve(x^2+x-1);
  evalf(solve(x^2+x-1));
```

$$-\frac{1}{2}+\frac{1}{2}\sqrt{5}, -\frac{1}{2}-\frac{1}{2}\sqrt{5}$$

$$0.6180339880, -1.618033988$$

Soustavy rovnic se také řeší přes příkaz *solve()*. Rovnice napíše do složených závorek. Máme-li výraz s více proměnnými, připíšeme do složených závorek, podle kterých proměnných se má soustava rovnic řešit.

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

$$\{x = 1, y = -1\}$$

Užitečný příkaz pro zjednodušení výrazu je *simplify()*.

```
> simplify(sin(x)^2+ln(2*y)+cos(x)^2);
```

$$1 + \ln(2) + \ln(y)$$

Pro počítání v **lineární algebře** si po otevření knihovny **LinearAlgebra** můžeme definovat vektory, matice a poté s nimi počítat. Pro vytvoření vektoru použijeme příkaz *Vector()* a pro vytvoření matice příkaz *Matrix()*. Několik příkladů zadání vektoru a matice.

```
> V1:=Vector([1, 0, 0]);
```

$$V1 := \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

```
> V2:=Vector[row]([1, 0, 0]);
```

$$V2 := [1, 0, 0]$$

```
> V3:=<1, 3, 5>;
```

$$V3 := \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$$

```
> V4:=<2 | 4 | 6>;
```

$$V4 := [2, 4, 6]$$

```
> A1:=Matrix(3,2,[[1,0],[0,1],[1,1]]);
```

$$A1 := \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

```
> A2:=<<1, 3, 5> | <7, 9, 11>>;
```

$$A2 := \begin{bmatrix} 1 & 7 \\ 3 & 9 \\ 5 & 11 \end{bmatrix}$$

```
>  A3:=<<0 |2 | 4 >, <6 | 8 | 10>>;
```

$$A3 := \begin{bmatrix} 0 & 2 & 4 \\ 6 & 8 & 10 \end{bmatrix}$$

Počítání s pojmenovanými vektory a maticemi je pak jednoduché.

```
> A1 + A2;
```

$$\begin{bmatrix} 2 & 7 \\ 3 & 10 \\ 6 & 12 \end{bmatrix}$$

```
> V3 . 3;
```

$$\begin{bmatrix} 3 \\ 9 \\ 15 \end{bmatrix}$$

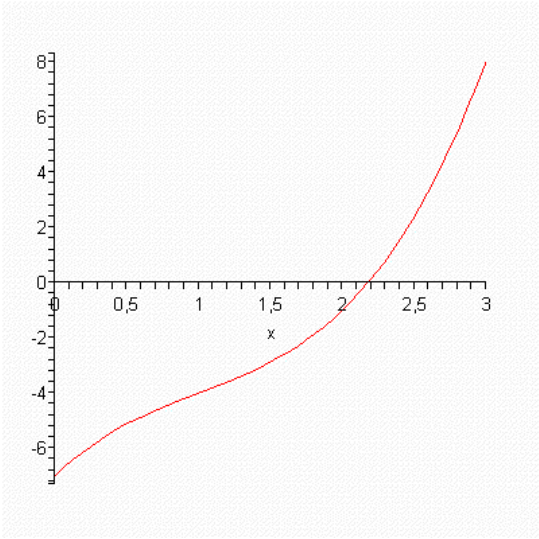
Pokud jde o řešení lineárních rovnic pomocí matic, pomůže nám při řešení příkaz *LinearSolve()*. Do matice M zapíšeme číselné hodnoty z levých stran rovnic podle stejného pořadí neznámých. Do matice N zapíšeme z pravých stran zbylé číselné hodnoty. Výslednou matici X dostaneme příkazem: *X:=LinearSolve(M, N);*

Zobrazování grafů

Velká část příkazů programu Maple je uložena v tzv. **knihovnách funkcí**. Chceme-li některý příkaz z této knihovny použít, musíme tuto knihovnu nejprve otevřít. Pokud budeme používat příkazů více, otevřeme knihovnu na začátku souboru použitím příkazu *with()*, př. *with(plots):*. Použijeme-li jen jeden příkaz, můžeme si funkci zavolat, aniž bychom otevírali knihovnu, přímo v místě, kde potřebujeme. Př. *plots[pointplot](A);* znamená zavolání funkce *pointplot()* s parametrem A z knihovny plots.

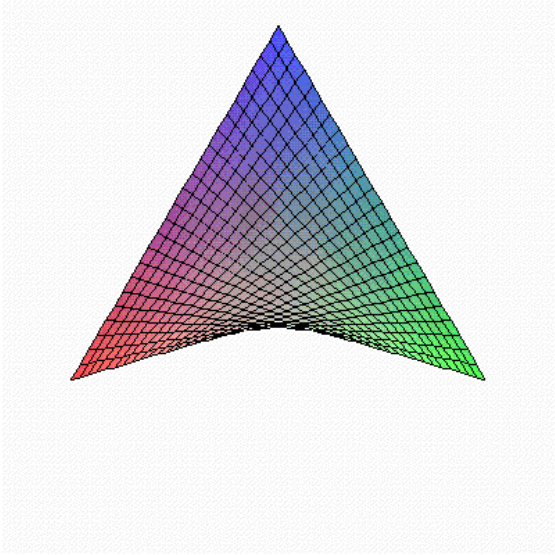
K zobrazování dat a matematických výrazů se používají **grafy**. Grafy vytvoříme jednoduché nebo komplexní, běžné nebo specializované, základní nebo uživatelský příjemné s reprezentací dat závisející na konkrétních příkazech a parametrech při tvorbě grafů. K vytvoření jednoduchého grafu **v rovině** (2-D grafu) se použije příkaz *plot()*, př. *plot(x^3-3*x^2+5*x-7, x=0..3);* kde hodnota x znázorňuje rozmezí na ose x, které se v grafu zobrazí.

```
> plot(x^3-3*x^2+5*x-7, x=0..3);
```



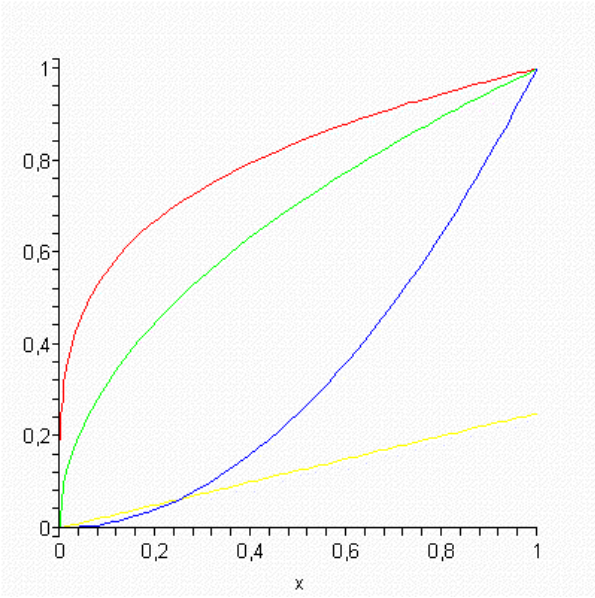
Pro zobrazení prostorového grafu (3-D graf) se napíše příkaz *plot3d()*. Například pro zobrazení funkce z=f(x,y)=xy se napíše do příkazového řádku následující text: *plot3d(x*y, x=-10..10, y=-10..10);* kde hodnoty pro x a y znamenají rozsah zobrazení po ose x a ose y.

```
> plot3d(x*y, x=-10..10, y=-10..10);
```



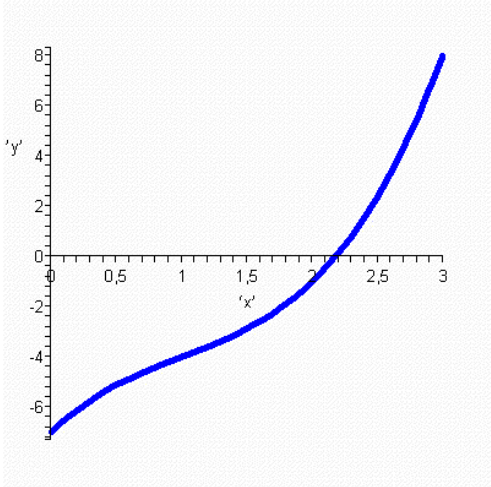
Chceme-li zobrazit více funkcí najednou, musíme tyto funkce zapsat do hranatých závorek, př. *plot([x^(1/4), sqrt(x), 1/4*x, x^2], x=0..1);*

```
> plot([x^(1/4), sqrt(x), 1/4*x, x^2], x=0..1);
```



Dalšího upravení grafu dosáhneme pomocí parametrů, které se píší za zobrazovanou funkci. Můžeme si změnit tloušťku čáry grafu pomocí parametru ‘*thickness*‘, barvu pomocí ‘*color*‘, popisky os pomocí ‘*labels*‘.

```
> plot(x^3-3*x^2+5*x-7, x=0..3, thickness=3, color=BLUE,
  labels=['x','y']);
```

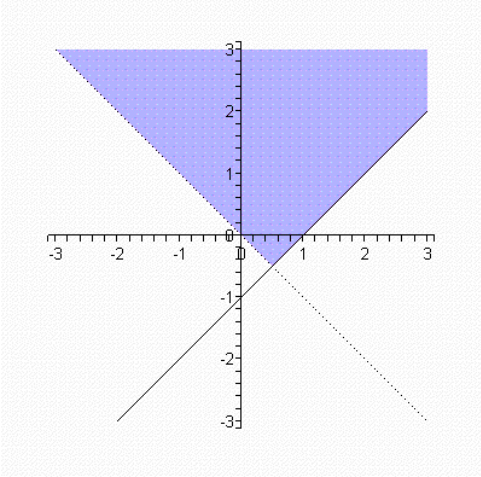


Požadujeme-li, aby měřítko osy x a osy y bylo stejné, připíšeme parametr *'scaling'* s hodnotou *'constrained'*. Název celého grafu získáme s příkazem *'title'*. Popis jednotlivých funkcí v grafu pomocí *'legend'*. Př. *plot([sin, cos], -Pi..Pi, title="Jednoduchý graf \n goniometrických funkcí", legend=[„Sinus", „Kosinus"]);*

Požadujeme-li zobrazení grafu funkce, která je zadána **implicitně**, tzn. ve tvaru např. x+y=1, použijeme příkaz *implicitplot()* z knihovny **plots**. Explicitně zadaná rovnice by vypadala například takto: x+y-1=0.

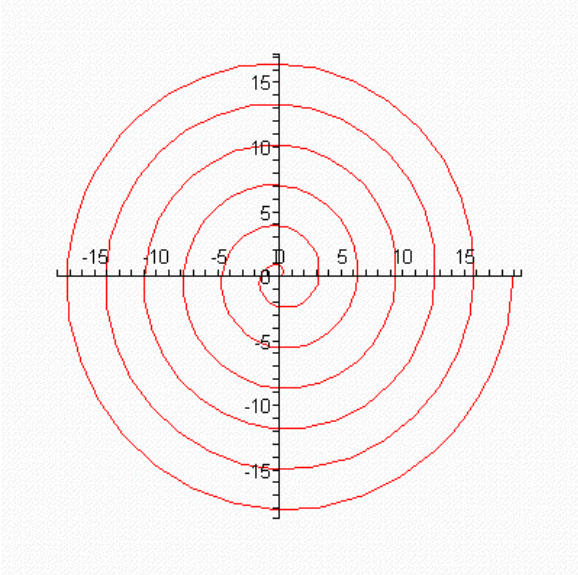
Př. *with(plots): implicitplots(x^2+y^2=1, x=-2..2, y=-2..2, thickness=3, color=blue, labels=['x','y'], scaling=CONSTRAINED);* Pro zobrazení nerovnic použijeme příkaz *inequal().*

```
> with(plots):
  inequal({x+y>0, x-y<=1}, x=-3..3, y=-3..3,
  optionsexcluded=(color=white, thickness=2));
```

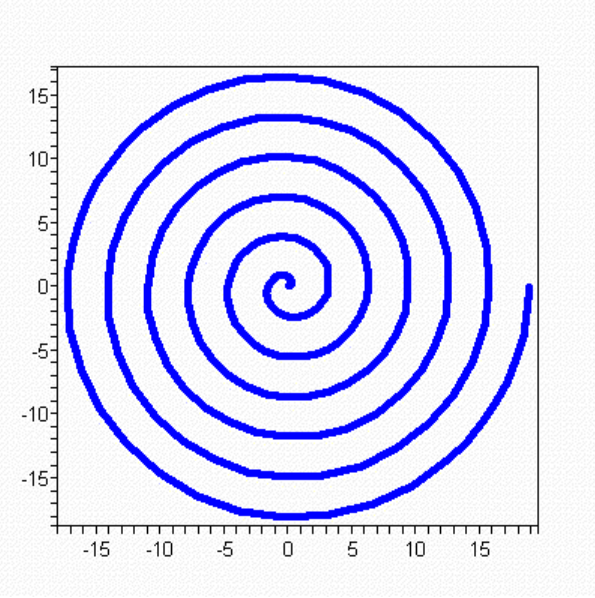


Zobrazovaná rovnice může mít také **parametrický tvar**. Chceme-li pak takovou funkci zobrazit, napíšeme příkaz *plot()*. Parametry pro zobrazení takto zadané funkce jsou podobné jako výše uvedené. Př. *plot([t*cos(2*t), t*sin(2*t), t=0..6*Pi]);*. Zobrazení grafu parametricky zadané křivky s danou hodnotu t bude s osovým křížem, *plot([t*cos(2*t), t*sin(2*t), t=0..6*Pi], thickness=3, color=BLUE, axes=BOXED, scaling=CONSTRAINED);*. Graf takto zadané křivky se zobrazí do čtverce se středem v počátku souřadnic.

```
> plot([t*cos(2*t), t*sin(2*t), t=0..6*Pi]);
```

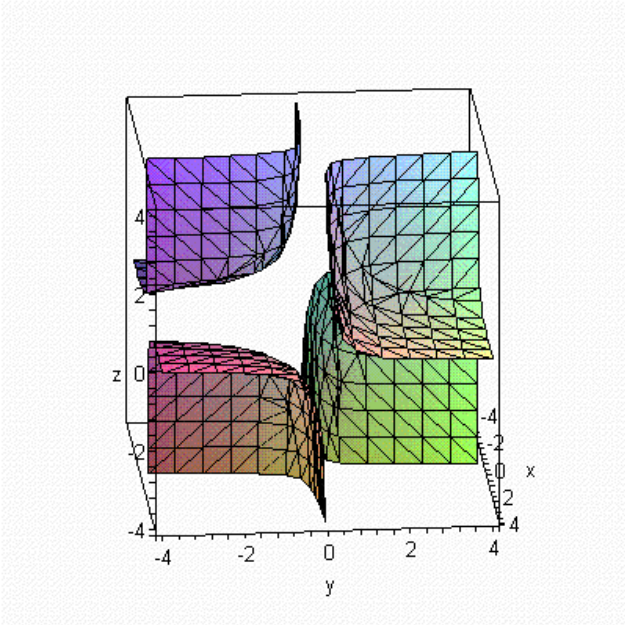



```
> plot([t*cos(2*t), t*sin(2*t), t=0..6*Pi], thickness=3,
  color=BLUE, axes=BOXED, scaling=CONSTRAINED);
```

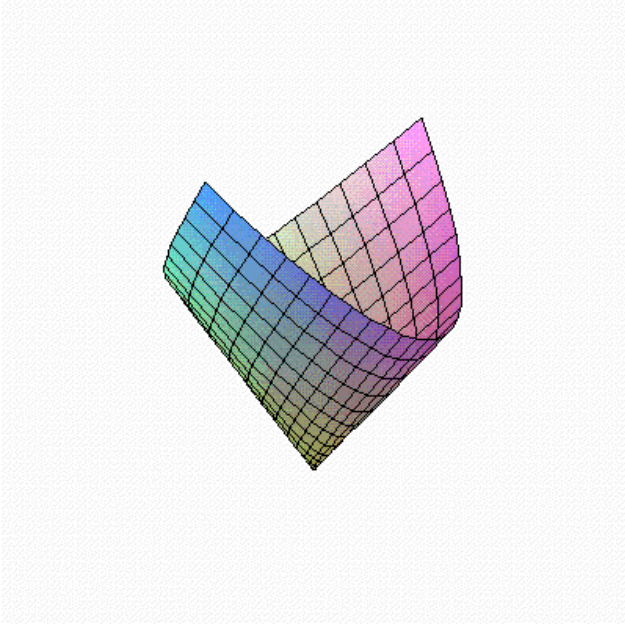


Zobrazování **prostorových (3-D) grafů** je podobné jako 2-D grafů. Základní zadání funkcí může mít opět několik tvarů. Běžný výraz, implicitní nebo parametrický tvar. Základní příkazy jsou potom *plot3d()*, *implicitplot3d()* a parametry jsou podobné jako u 2-D grafů.

```
> restart:
  with(plots):
  implicitplot3d(x*y*z=1, x=-4..4, y=-4..4, z=-4..4,
   axes=BOXED, scaling=CONSTRAINED, grid=[13, 13, 13]);
```




```
> restart:
  with(plots):
  plot3d([t*s, t^2-s^2, t^2+s^2], t=-2..2, s=-2..2);
```



Procedury

Jak již bylo výše popsáno, velká část příkazů je uložena v knihovnách. Nejsou tam ale vždy všechny, které bychom potřebovali. Pokud chceme použít příkaz, který není v žádné knihovně, můžeme si ho napsat sami pomocí procedur. Každá taková procedura musí mít svůj název a musí začínat a končit příkazem *proc*. Příklad takového naprogramovaného příkazu je funkce x^3. Procedura bude mít jeden parametr. Tím se stane číslo, které vstupuje do procedury. Ta s ním pak bude dále počítat. Při volání této procedury musíme parametr zadat. Procedura pak bude vypadat následovně: *g:= proc(x) x^3 end proc;* Volání pro zvolený parametr (číslo 2) je následující:

```
> g(2);
```

8

V případě, že je procedura složitější, necháváme spolu na řádku související příkazy. Jinak odřádkujeme, aby byl kód přehledný.

Cyklus IF-ELSE

V procedurách můžeme v případě potřeby použít cykly a příkazy známé z programování. Jedním takovým cyklem je **if-else**. Znamená když je splněna podmínka udělej toto a pokud splněna není udělej tamto. Použijeme ho pro naprogramování funkce x^2, která se vypočte jen pro vstupující hodnoty větší než 0. Jinak se nevypočte a napíše se jako výsledek 0. Příklad:

```
h:=proc(x)
    if x>0 then
            x^2
    else
            0
    end if;
 end proc;
```

Volání této funkce je pak stejné jako u té předcházející.

```
> h:=proc(x) if x>0 then x^2 else 0 end if; end proc:
> h(-5);
                                  0
> h(5);
                                  25
```

Cyklus WHILE-DO

Posledním typem cyklů je **while-do**. Cyklus bude provádět příkazy tak dlouho, dokud bude splněna podmínka. Jakmile podmínka splněna nebude, bude cyklus ukončen.

```
pocitej:=proc(x,y)
    local z;
    z := evalf (x+2*y);
    while abs(z) < 10 do
            z := z^2;
    end do;
  end proc;
```

Daná procedura spočítá pro dvě předem zadaná čísla x a y druhou mocninu hodnoty z funkce z=x+2*y. A to pouze v případě, že funkční hodnota bude v intervalu (-10; 10).

FOR cyklus

Další typ je **for cyklus**. Tento cyklus se provádí dle zadaného počtu opakování. Při každém průchodu cyklu se provedou dané příkazy. Příkladem může být výpis různých úhlů.

```
uhel:=proc(n)
    for i to n do
            Uhel[i]:=2*Pi*i/n
    end do:
  end proc;
```

Cyklus TRY-CATCH

Dalším typem je **try-catch** cyklus. Cyklus try bude provádět příkazy. Pokud se však vyskytne výsledek zapsaný v příkazu catch, zachytí se a provede se příkaz popsaný v této větvi cyklu. Příkladem bude cyklus, který provádí danou metodu, pokud její výsledek nebude FAIL. V tom případě se provede jiná metoda.

```
zkouska:=proc()
    try
            result:=MethodA(f,x)
    catch ”FAIL”:
            result:=MethodB(f,x)
    end do:
 end proc;
```

Procedury je nutné použít, pokud potřebujeme například určit speciální funkci tlačítka. V mapletu se již procedura naprogramovat nedá. Proto se všechny procedury musejí naprogramovat nejdříve. V mapletu je můžeme jen vyvolávat. Vyvoláme je většinou s parametrem (parametry). Parametrem je hodnota některé komponenty, kterou zadal uživatel mapletu.

Procerura napsaná před samotným mapletem:

```
Polomer := proc(rovnice)
  circle(k,rovnice,[x,y]);
  evalf(radius(k));
 end proc:
```

Podrobnější popis volání procedury a další používání bude uveden dále.

4.3 Tvorba apletu v Maple (maplet)

Aplet je grafické uživatelské rozhraní, které se podobá oknu, umožňující zpracovávat nebo ovládat některá data. Aplikace vytvořená v programu Maple se nazývá maplet. Spouštět se poté může z programu Maple (přípona souboru bude .mw nebo .mws) nebo zcela samostatně ze souboru s příponou .maplet. Příkladem takového mapletu může být aplikace, kde uživatel zadá libovolné číslo a aplikace uživateli zodpoví otázku *Je zadané x kladné?*

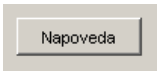
Autoři mapletů píší různé definice a jiný potřebný kód v Maple. Uživatelé mapletů je mohou jednoduše používat, aniž by museli znát jakékoli příkazy a definice tohoto programu.

Příkazy, které budeme psát, je dobré vhodně členit. Nové příkazy psát na novou řádku. Musíme udržovat dobrou čitelnost celého kódu hlavně pro případ, kdy bude tento kód po nás někdo číst.

Seznam komponent pro možné použití v mapletu

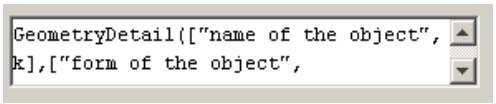
Texty

Tlačítka (Buton)

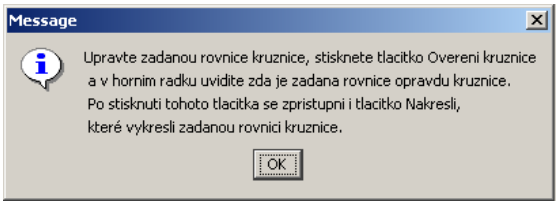


Vstupní textové pole (TextField)

Výstupní textové pole (TextBox)

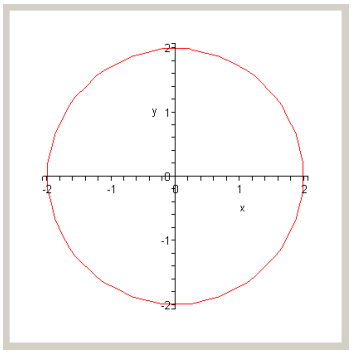


Nápověda (MessageDialog)



Posuvník (Slider)

Okno grafů (Plotter)



Zaškrtávací políčko (CheckBox)

Vybírací políčko (RadioButton)

Seznam položek (ComboBox)

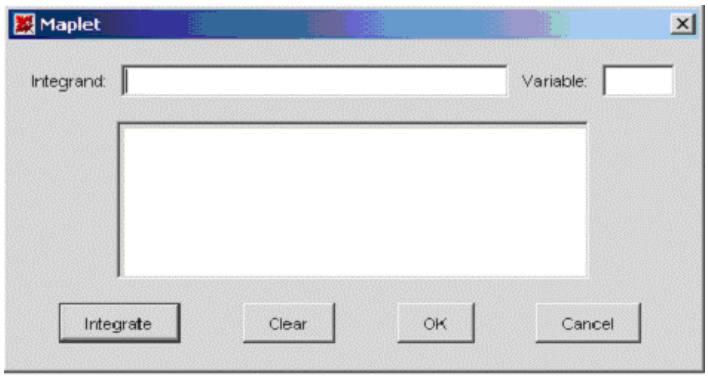
Knihovny

Pro psaní a spouštění mapletů se využívá knihovna funkcí s názvem **Maplets**. Tato knihovna je rozdělena na dvě části. První část obsahuje nástroje pro tvorbu a spouštění apletů. Druhá obsahuje již hotové příklady vytvořených mapletů. Knihovna **Maplets** obsahuje tyto skupiny funkcí a nástroje: **Display, Elements, Tools.** Jednotlivé podknihovny se zapisují ve tvaru hlavní knihovna[podknihovna].

Knihovna **Maplets[Display]** obsahuje funkce k zobrazení a spuštění mapletů. Knihovna **Maplets[Elements]** obsahuje funkce pro definování jednotlivých komponent neboli elementů neboli základních použitých součástí v mapletu. Příkladem těchto součástí můžou být různá okna, textová pole, popisky, tlačítka, zaškrtávací políčka, přepínací políčka, menu a další.

Knihovna **Maplets[Tools]** obsahuje nástroje pro pomoc autorům mapletů při jejich práci na vývoji mapletů. Pro zobrazení více informací o těchto podknihovnách stačí zadat příkaz *?Maplets[Tools]*, případně zadáme název jiné podknihovny.

Druhá podknihovna knihovny Maplets se jmenuje **Maplets[Examples]** a obsahuje již hotové jednoduché příklady. Příkladem takovéhoto příkladu je tento obrázek.



Na tomto mapletu si můžeme vyzkoušet integrování zadaného výrazu podle dané proměnné. Do pole ***Integrand*** zapíšeme výraz pro integrování. Do pole ***Variable*** zapíšeme, podle které proměnné se bude integrovat. A do pole pod prvním řádkem dostaneme po stisknutí tlačítka ***Integrate*** výsledek. Tlačítko ***Clear*** smaže všechny zadané hodnoty, tlačítka ***OK*** a ***Cancel*** zavřou okno.

Jednoduchý maplet

Vytvoření jednoduchého mapletu obsahující pouze text „Dobrý den!“ není nijak těžké. Potřebujeme zavolat knihovnu s prvky **Maplets[Elements]**, definovat název mapletu a maplet jako takový, nebo-li okno, které se nám otevře.

```
> with(Maplets[Elements]):
  muj_maplet := Maplet ( [[ "Dobry den!" ]]):
  Maplets[Display]( muj_maplet );
```

Abychom předešli chybám, způsobeným při zpracování mapletu, je vhodné, nepoužívat diakritiku. Maplet je nazván jako muj_maplet. Názvu mapletu je přiřazena funkce *Maplet()*, která definuje maplet jako takový, neboli okno, které chceme otevřít. Parametrem této funkce je vše, co bude obsahovat. Uvnitř mapletu je jedna řádka s jedním textem. Počty řádků a jednotlivých elementů řazených vedle sebe určují hranaté závorky []. Všechny texty, které chceme zapsat se uvádějí v uvozovkách „“. Opět je příkaz ukončen středníkem. Abychom vůbec vytvořený maplet mohli vidět, musíme si jej

zobrazit. K tomu budeme potřebovat knihovnu **Maplets[Display]**. Aby knihovna věděla, co chceme zobrazit, uvedeme jako parametr název mapletu, který má být zobrazen.

Maplet má předem předvolený název neboli titulek „Maplet“. Obsahuje v pravém horním rohu tlačítko s křížkem k zavření celého mapletu. Pro nastavení jiného titulku mapletu budeme muset použít jiný příkaz. Vytvoříme ho příkazem *Maplet()*, ale v mapletu vytvoříme okno příkazem *Window()*. Parametry pak budou název titulku a prvky mapletu.

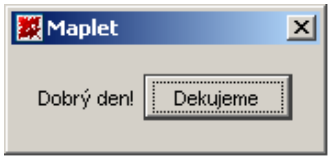
```
> muj_maplet := Maplet (  Window( "Muj prvni maplet", [
  "Dobry den!",
  "Toto je muj prvni maplet!"] ) ) :
  Maplets[Display]( muj_maplet );
```

Tento maplet bude mít dané texty pod sebou a název okna bude „Muj prvni maplet“.

Tlačítka

K našemu již vytvořenému mapletu přidáme tlačítko s názvem „Děkujeme“. To nebude dělat nic jiného, než že po stisknutí zavře celý maplet. Příkaz pro vytvoření tlačítka je *Button()*. Má několik parametrů. Můžeme definovat název tlačítka a akci, která se provádí při jeho stisknutí. Pro náš příklad bude stačit napsat tento příkaz: *Button(„Děkujeme“, Shutdown());* Tedy název se opět píše do uvozovek a funkce pro zavření mapletu se jmenuje *Shutdown()*.

```
> with(Maplets[Elements]):
  muj_maplet := Maplet ( [
  [ "Dobry den!", Button("Dekujeme", Shutdown()) ]
  ] ) :
  Maplets[Display]( muj_maplet );
```



Při takovémto tvaru příkazu, bude maplet vypadat tak, že texty i tlačítko budou vedle sebe. Pokud bychom potřebovali mít v mapletu text a tlačítko pod sebou, musel by příkaz vypadat následovně.

```
> with(Maplets[Elements]):
  muj_maplet := Maplet ( [
  [ "Dobry den!"],
  [Button("Dekujeme", Shutdown()) ]
  ] ) :
  Maplets[Display]( muj_maplet );
```

Pokud bychom chtěli použít konstrukci mapletu s titulkem, daný příkaz by musel vypadat následovně:

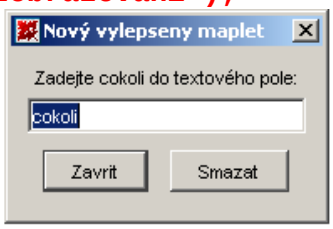
```
> muj_maplet := Maplet ( Window( "Maplet s tlacitkem", [
    "Dobry den!",
  Button("Dekujeme", Shutdown())
  ] ) ) :
  Maplets[Display]( muj_maplet );
```

Funkce *Shutdown()* může mít i parametr. Byl by jím název další součásti mapletu a výsledkem by byla hodnota této součásti, která se nám zobrazí po zavření mapletu v Maple. Jinou akci, kterou může tlačítko provést je nastavení některé jiné části mapletu. Příklad si ukážeme až u některých dalších součástí mapletu.

Vstupní textové pole

Vstupní textové pole je pole, do kterého může uživatel zapisovat texty, čísla, popřípadě různé znaky. Příkaz pro vytvoření je *TextField[TF1]()*. Do hranatých závorek píšeme název této jediné komponenty tak, aby byl jednoznačný. Hlavně v případě, že máme v mapletu více komponent této skupiny. Název je též nutný pro následné počítání či pracování s hodnotou, kterou uživatel do tohoto pole zadá. Jako parametr může například zadat text, který se nám pak zobrazí po otevření mapletu.

```
> restart:
  with(Maplets):
  zobrazovani := Maplet ( Window( "Maplet pro zobrazovani
  primky", [
  ["Zadejte rovnici primky:",TextField [TF1] (20, "cokoli")],
  [ Button ("Zavrit", Shutdown(TF1) ),
    Button("Smazat", Action( SetOption( TF1 = "" ) ))]
   ] )):
  Maplets[Display]( zobrazovani );
```

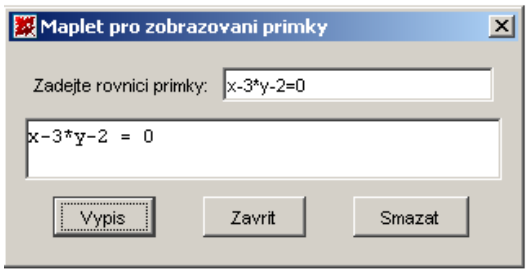


Maplet obsahuje nám již známé části jako titulek, vložený text a nyní již známé textové pole. Toto pole má jako parametr text „cokoli“, který se nám zobrazí po otevření mapletu. Pod tímto polem jsou dvě tlačítka seřazena vedle sebe. To jsme zajistili zapsáním těchto tlačítek do společných hranatých závorek. První tlačítko nám maplet zavře. To zajišťuje funkce *Shutdown()* s parametrem TF1, tzn. že po zavření mapletu se nám do Maple zapíše hodnota z textového pole. Pokud do textového pole uživatel nic nezadá, vypíše se „cokoli“ a pokud text v textovém poli uživatel změní, bude výstupem tohoto mapletu právě hodnota zadána uživatelem. Druhé tlačítko se jmenuje smazat a měla by zajistit smazání jakéhokoli textu v textovém poli. To nám zajišťuje funkce *SetOption()*, která mění obsahy zadaných elementů mapletu. My máme zadaný element TF1. To je název pro naše vstupní textové pole. Tento element nastavujeme na „“, tzn. že v textovém poli nebude žádný text. Uvozovky jsou uvozujícím znakem pro texty.

Výstupní pole

Někdy potřebujeme uživateli sdělit nějakou zprávu, ukázat jím zapsaný text apod. Pro tyto účely nám slouží výstupní textové pole. Zadává se příkazem *TextBox[TB1]()*. Opět musíme zadat jednoznačné pojmenování daného elementu a to např. TB1. Parametr, který můžeme zvolit je možnost zapisovat do tohoto pole. Tento parametr volíme příkazem *'editable'*. Může nabývat dvou hodnot. Buď povolíme zapisovat, hodnota tedy bude true, nebo zápis zakážeme a hodnota bude false. Dále můžeme zvolit, jak velké toto pole bude. Volíme počet řádků a počet znaků na řádku. Pokud by se nám do tohoto pole zobrazovaný text nevešel, automaticky se po pravé straně zobrazí roletka, kterou se dostaneme dále na konec zobrazovaného textu.

```
> restart:
  with(Maplets):
  zobrazovani := Maplet ( Window( "Maplet pro zobrazovani
  primky", [
  [ "Zadejte rovnici primky: ",  TextField [TF1] (20) ],
  TextBox[TB1](editable=false, 2..20),
  [ Button ("Vypis", Action( Evaluate(TB1= 'TF1')  )     ),
    Button ("Zavrit", Shutdown(TB1) ),
    Button("Smazat", Action( SetOption( TF1 = "" ), SetOption(
    TB1 = "" ) ))
  ]
  ] )    ) :
  Maplets[Display]( zobrazovani );
```



V mapletu jsme použili hranaté závorky pro uspořádání více prvků na řádek. Vstupní textová pole TF1 mají jako parametr číslo 20, tzn. že textové pole bude velké 20 znaků. Pod vstupní textová pole jsem umístila výstupní textové pole se dvěma řádky o délce 20 znaků. Výstupní textové pole není přepisovatelné. Pod výstupním polem jsou vedle sebe tři tlačítka. První tlačítko ***Výpis*** obsahuje funkci *Evaluate()* s jedním parametrem. Tato funkce přiřazuje vybranému elementu další funkci. V našem případě se do výstupního

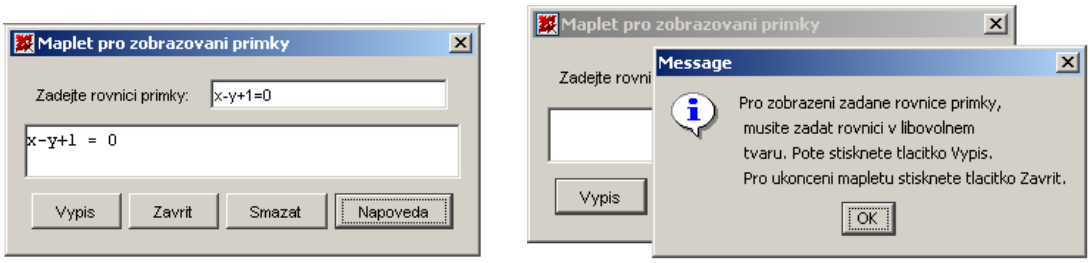
textového pole zobrazí zadaná rovnice z textového pole. Druhé tlačítko ***Zavřít*** již známe. Výstupem bude vypsání obsahu výstupního pole po uzavření mapletu. Poslední tlačítko ***Smazat*** nám vymaže texty ze všech textových elementů mapletu. Pomůže nám k tomu funkce *Action()*, jejímž parametry jsou tři funkce *SetOption()* a to pro každé ze tří textových polí v mapletu.

Nápovědy

Každý maplet by měl obsahovat nápovědu pro ovládání daného mapletu. Tu si vyvoláme stiskem tlačítka ***Napoveda***. Poté se nám otevře další okno, obsahující text nápovědy. Tento text zapíšeme pomocí příkazu *MessageDialog[MD1]()*. Parametrem je v uvozovkách napsaný text nápovědy a určení typu dialogu. Pro naši nápovědu použijeme typ '*information*'. Dialog se ovšem nezapisuje přímo do okna mapletu mezi ostatní pole a tlačítka, ale až za ukončující závorku funkce *Windows()*. Aby se nám tato nápověda zobrazila, musíme správně nastavit funkčnost daného tlačítka. Vyvolání dialogu způsobí funkce *RunDialog()*, kde parametrem je název daného dialogového textu.

```
> restart:
  with(Maplets):
  zobrazovani := Maplet ( Window( "Maplet pro zobrazovani
  primky", [
  [ "Zadejte rovnici primky: ",  TextField [TF1] (20) ],
   TextBox[TB1](editable=false, 2..20),
  [ Button ("Vypis", Action( Evaluate(TB1= 'TF1')  )     ),
    Button ("Zavrit", Shutdown(TB1) ),
    Button("Smazat", Action( SetOption( TF1 = "" ), SetOption(
     TB1 = "" ) )),
    Button ("Napoveda", RunDialog(MD1) )    ]
   ] ) ,
  MessageDialog[MD1](" Pro zobrazeni zadane rovnice primky,\n
  musite zadat rovnici v libovolnem\n tvaru. Pote stisknete
  tlacitko Vypis. \n Pro ukonceni mapletu stisknete tlacitko
  Zavrit. ", 'information')    ) :

  Maplets[Display]( zobrazovani );
```

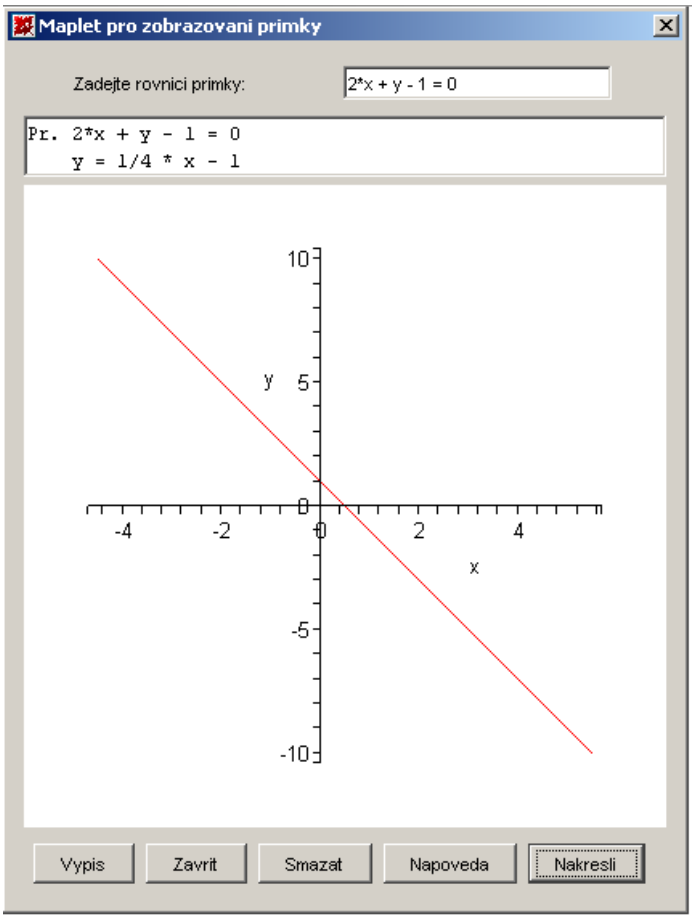


Stiskem tlačítka Nápověda se aktivuje funkce *RunDialog()* s parametrem názvu daného dialogu MD1. Dalšími typy dialogu kromě informativního může být upozornění, error nebo bez udání typu dialogu.

Zobrazování grafů

Pro zobrazení různých grafických výstupů potřebujeme okno grafů, ve kterém se grafické výstupy zobrazí. Dané okno grafů vložíme do mapletu příkazem *Plotter()*. Grafické objekty se sice zobrazí v okně grafů, ale vždy až po stisku tlačítka ***Nakresli***, pokud je správně nastaveno. Připravený maplet bude fungovat tak, že po výpisu zadaného bodu (po stisku tlačítka ***Vypis***) se nám zpřístupní i tlačítko ***Nakresli***. Dokud tlačítko ***Výpis*** nebude stisknuto, nebude nám zpřístupněno ani tlačítko ***Nakresli***, tzn. že bude šedé a bude nefunkční.

```
> restart:
  with(plots):
  with(geometry):
  with(Maplets):
  zobrazovani := Maplet ( Window( "Maplet pro zobrazovani
  primky", [
  [ "Zadejte smernici primky: ",  TextField [TF1] (3) ],
   TextBox[TB1](editable=false, 2..20, "y = (zadana
    smernice)*x + posunuti "),
   Plotter[PL1]( plot(undefined, x=-5..5)),
  [ Button ("Vypis", Action( Evaluate(TB1= 'y = TF1*x + SL1')
     )     ),
    Button ("Zavrit", Shutdown(TB1) ),
    Button("Smazat", Action( SetOption( TF1 = "" ),
      SetOption( TB1 = "" ),
      SetOption( SL1 = 0 )  )    ),
    Button ("Napoveda", RunDialog(MD1) ),
    Button[B5]("Nakresli", Evaluate( PL1 = 'implicitplot(y =
      TF1*x + SL1, x=-10..10, y=-10..10)') ) ]   ] ) ,
  MessageDialog[MD1]("Pro zobrazeni zadane rovnice primky,\n
   musite zadat rovnici v libovolnem\n tvaru. Pote stisknete
   tlacitko Vypis. \n Pro ukonceni mapletu stisknete tlacitko
   Zavrit.", 'information')    ) :
  Maplets[Display]( zobrazovani );
```



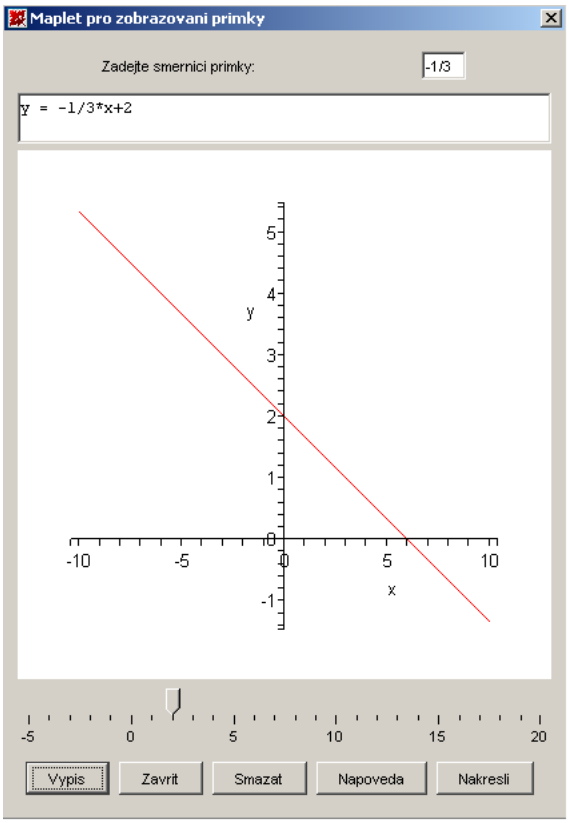
Do mapletu jsme oproti poslední verzi vložili okno grafů PL1. V daném okně nebude zobrazeno nic, to nám určuje parametr *'undefinated'*. Zobrazení osy x bude v intervalu <-5, 5>. Dále jsme upravili funkčnost tlačítka ***Výpis***. Abych mohla nastavit viditelnost tlačítka ***Nakresli*** v závislosti na tlačítku ***Výpis***, potřebovala jsem do tlačítka ***Výpis*** vložit funkci *Action()*. Jako její další parametr nastavit funkci *SetOption()* na funkčnost či nefunkčnost závislého tlačítka Nakresli. A na závěr jsme vložili další tlačítko označené jako B5 s názvem ***Nakresli***. Bude zobrazovat v okně grafů hodnotu z výstupového textového pole TB1.

Posuvník

Pro interaktivní změnu grafu při změně některých parametrů nakresleného grafu nám slouží posuvník. Pro zjednodušení, je to měřítko, které má na sobě jakýsi zobáček, který lze pomocí myši posouvat v horizontálním směru. Příkaz pro vytvoření posuvníku je *Slider[SL1]()*. Má parametry rozsah od čísla do čísla zadaný pomocí dvou teček mezi čísly, nastavenou výchozí hodnotu, případně další nastavení.

```
> restart:
  with(plots):
  with(geometry):
  with(Maplets):
  zobrazovani := Maplet ( Window( "Maplet pro zobrazovani
  primky", [
  [ "Zadejte smernici primky: ",  TextField [TF1] (3) ],
    TextBox[TB1](editable=false, 2..20, "y = (zadana
     smernice)*x + posunuti "),
    Plotter[PL1]( plot(undefined, x=-5..5)),
    Slider[SL1](-5..20, 0, 'majorticks' = 5, 'minorticks' = 1,
                    Evaluate(PL1 = implicitplot(y=TF1*x+SL1,
       x=-10..10, y=-10..10)')   ),
  [ Button ("Vypis", Action( Evaluate(TB1= 'y = TF1*x + SL1')
      )     ),
    Button ("Zavrit", Shutdown(TB1) ),
    Button("Smazat", Action( SetOption( TF1 = "" ),
      SetOption( TB1 = "" ), SetOption( SL1 = 0 ) )  )    ),
    Button ("Napoveda", RunDialog(MD1) ),
    Button[B5]("Nakresli", Evaluate( PL1 = 'implicitplot(y =
      TF1*x + SL1, x=-10..10, y=-10..10)') )    ]
  ] ) ,
  MessageDialog[MD1](" Pro zobrazeni zadane rovnice primky,\n
    musite zadat rovnici v libovolnem\n tvaru. Pote stisknete
    tlacitko Vypis. \n Pro ukonceni mapletu stisknete tlacitko
    Zavrit.", 'information')    ) :
  Maplets[Display]( zobrazovani );
```

Do vstupního textového pole TF1 budeme zadávat směrnici k přímky y=k*x+q. Ve výstupním textovém poli TB1 se nám po stisknutí tlačítka ***Vypis*** se bude zobrazovat rovnice aktuálně nastavené přímky. Výchozí text, který se zobrazí při otevření mapletu bude „y = (zadaná směrnice) * x + (posunutí)“. Po stisknutí tlačítka Nakresli nebo při jakékoli změně hodnoty na posuvníku, uvidíme v okně grafů PL1 graf dané přímky.

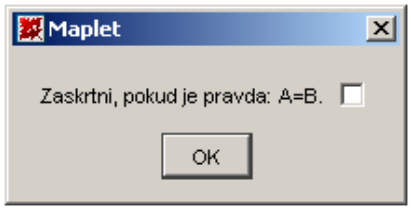


V okně grafů PL1 uvidíme graf dané přímky a to po stisknutí tlačítka ***Nakresli*** nebo při jakékoli změně hodnoty na posuvníku. Pod oknem grafů je posuvník SL1, kde zvolíme q, nebo-li posunutí přímky. Volbu q nastavíme na posuvníku v rozmezí <-5, 20> a výchozí hodnotou je 0. Při jakékoli změně hodnoty na posuvníku, se v okně grafů znova zobrazí dané přímka tvořená zadanou směrnicí a zadaným posunutím přímky. Parametrem *‘majorsticks‘* nastavujeme číslování rozděleného posuvníku po větších celcích. Parametrem *‘minorsticks‘* nastavujeme nejmenší jednotku, po které je posuvník rozdělen. Je nastavena funkce *Action()* tak, aby se při každé změně nakreslil graf v okně grafů znovu. Po stisknutí tlačítka ***Výpis***, se zobrazí ve výstupním textové poli rovnice přímky, která je aktuálně nastavena pomocí směrnice přímky a posunutí přímky. Do tlačítka ***Smazat*** jsem připsala upravení hodnoty posuvníku na nulovou hodnotu, smazání grafu přímky a vymazání všech hodnot z textových polí. Tlačítko ***Nakresli*** jsme pozměnili tak, aby se mám zobrazovala v okně grafů zadaná přímka. Ostatní elementy mapletu jsme ponechali beze změny vůči předchozímu mapletu.

Zaškrtávací políčko

Toto čtvercové políčko nabývá hodnot pravda nebo nepravda – výstupem bude true nebo false. Zobrazíme ho pomocí příkazu *CheckBox[ChB1]()*. Parametrem může být výchozí nastavení hodnoty pomocí value na hodnotu true nebo false. V případě, že máme více těchto zaškrtávacích políček, můžeme jejich hodnotu libovolně nastavovat na hodnoty true nebo false - tedy vyplněno, nevyplněno.

```
> Maplets[Display] ( Maplet ( [
        [ "Zaskrtni, pokud je pravda: A=B." ,
        CheckBox [ChB1] ( value = false ) ],
        Button ("OK", Shutdown([ChB1]) )
   ] ) ) ;
```

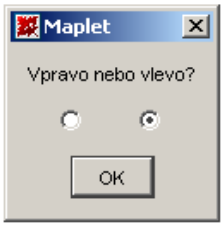


Maplet jsem zapsali přímo jako parametr zobrazovací funkce mapletu.

Vybírací políčko

Také toto políčko nabývá hodnot true nebo false. Rozdíl mezi předchozím a tímto políčkem je ten, že u více zobrazených polí patřících do jedné skupiny si můžeme vybrat pouze jedno vybírací políčko ze všech. Zobrazíme jej příkazem *RadioButton[RB1]()* a parametrem může být opět předvolená hodnota true nebo false.

```
Maplets[Display] ( Maplet ( [
         "Vpravo nebo vlevo?" ,
        [RadioButton [RB1] ( false, group = BG1 ) ,
         RadioButton [RB2] ( true, group = BG1 ) ] ,
        Button ("OK", Shutdown([RB1, RB2]) )
        ] ,
        ButtonGroup[BG1]()   )   ) ;
```

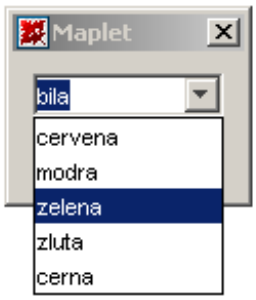


Každé vybírací políčko RB1 i RB2 jsem zařadila do skupiny BG1, kterou jsem definovala na konci mapletu. Pokud bychom políčka do skupiny nezařadili, mohli bychom si zaškrtávat hodnoty políček libovolně jako u zaškrtávacích políček.

Seznam položek

Seznam položek svým tvarem připomíná vstupní textové pole. Rozdíl je ale v tom, že seznam položek je rozbalovací seznam, který se nám rozvine pouze při kliknutí na pravou část seznamu se šipkou. Zobrazíme jej příkazem *ComboBox[CoB1]()*. Parametry tohoto seznamu položek je nastavení výchozí položky, kterou uvidíme, i když seznam nerozvineme, a seznam všech položek, které budou schované v seznamu.

```
> Maplets[Display] ( Maplet ( [
        ComboBox [CoB1] ( 'value' = "bila",
  ["cervena", "modra", "zelena", "zluta", "cerna"] )  ,
        Button ("OK", Shutdown([CoB1] ) )
        ]    )   );
```



Po spuštění mapletu se nám v řádku zobrazí bílá barva. Při kliknutí na tento řádek se nám rozbalí roletka, kde si ze seznamu barev můžeme vybrat námi požadovanou barvu. Pokud na tuto barvu klikneme, zobrazí se nám do řádku místo původně nastavené barvy. Stisknutím tlačítka OK se nám maplet zavře.

4.4 Popis vytvořeného mapletu

```
restart;
with (Maplets[Elements]):
with(plots):
maplet:=
  Maplet(
     Window("Vzajemna poloha plochy a roviny",
       [[[
         ["Zadejte rovnici plochy:", TextField[TF1](15)],
        "Zadejte koeficienty a, b, c, d roviny:",
         ["a", Slider[SL1]( -15..15, 1,
Evaluate('PL1'='implicitplot3d([TF1,
SL1*x+SL2*y+SL3*z+SL4=0])' ), 'onchange'=
Action(Evaluate('PL1'='implicitplot3d([TF1,
SL1*x+SL2*y+SL3*z+SL4=0], x=-10..10, y=-10..10,z=-10..10,
labels=[x,y,z],  scaling='CONSTRAINED', numpoints=2000,
color=[blue, red], axes=frame)')),
                          'showticks', 'majorticks'=5,
'minorticks'=1)],

      ["b", Slider[SL2]( -15..15, 1,
Evaluate('PL1'='implicitplot3d([TF1,
SL1*x+SL2*y+SL3*z+SL4=0])' ), 'onchange'=
Action(Evaluate('PL1'='implicitplot3d([TF1,
SL1*x+SL2*y+SL3*z+SL4=0], x=-10..10, y=-10..10,z=-10..10,
labels=[x,y,z],  scaling='CONSTRAINED', numpoints=2000,
color=[blue, red], axes=frame)')),
                          'showticks', 'majorticks'=5,
'minorticks'=1)],

       ["c", Slider[SL3]( -15..15, 1,
Evaluate('PL1'='implicitplot3d([TF1,
SL1*x+SL2*y+SL3*z+SL4=0])' ), 'onchange'=
Action(Evaluate('PL1'='implicitplot3d([TF1,
SL1*x+SL2*y+SL3*z+SL4=0], x=-10..10, y=-10..10,z=-10..10,
labels=[x,y,z],  scaling='CONSTRAINED', numpoints=2000,
color=[blue, red], axes=frame)')),
                          'showticks', 'majorticks'=5,
'minorticks'=1)],

       ["d", Slider[SL4]( -15..15, 1,
Evaluate('PL1'='implicitplot3d([TF1,
SL1*x+SL2*y+SL3*z+SL4=0])' ), 'onchange'=
Action(Evaluate('PL1'='implicitplot3d([TF1,
SL1*x+SL2*y+SL3*z+SL4=0], x=-10..10, y=-10..10, z=-
10..10,labels=[x,y,z],  scaling='CONSTRAINED',
numpoints=2000,  color=[blue, red], axes=frame)')),
                          'showticks', 'majorticks'=5,
'minorticks'=1)]],

Plotter[PL1](implicitplot3d(0*x+0*y+0*z=1, x=-10..10,y=-
10..10,z=-10..10, labels=[x,y,z], scaling='CONSTRAINED',
numpoints=2000) )],
```

```
        [ TextBox[TB1]('editable'='false', 1..30),

Button['B5']("Vykresli",
Evaluate('PL1'='implicitplot3d([TF1,
SL1*x+SL2*y+SL3*z+SL4=0],  x=-10..10,y=-10..10, z=-
10..10,numpoints=2000, color=[blue, red], axes=frame)')),
          Button ("Vypis", Action( Evaluate(TB1=
'SL1*x+SL2*y+SL3*z+SL4=0')  )     ),
          Button("Vycistit",
                   Action( SetOption(TF1=""),
                           SetOption(TB1=""),
                           Evaluate(PL1='plot3d([0,0,0],
x=-10..10,y=-10..10,z=-10..10)')
                         )
                ),
        Button("OK", Shutdown([TF1, SL1, SL2, SL3, SL4])),

          Button("Napoveda",RunDialog(MD1))
         ]
        ]
      ),
     MessageDialog[MD1](" Pro zobrazeni vzajemne polohy
obou rovnic musite zadat rovnici prvni roviny! \n Pote
stisknete tlacitko Nakresli a obe roviny se zobrazi v okne
grafu. Pro vypsani rovnice \n druhe roviny  stisknete
tlacitko Vypis Pro ukonceni mapletu stisknete tlacitko
Zavrit.
", 'information')
 ):

Maplets[Display](maplet);
```

(1) Příkazem vymažeme z paměti všechny proměnné a jim přiřazené hodnoty.

(2)(3) otevřeme knihovny Maple pro používání dalších funkcí z těchto knihoven.

(4) jménu maplet přiřazujeme funkci (5) z otevřené knihovny.

(6) daný maplet vytvoříme pomocí okna (6), které bude mít daný titulek

(7) určujeme umístění prvků v mapletu (dle umístění ukončujících závorek)

(8) popisek a vedle je textové pole o velikosti 15 znaků

(9) popisek

(10) popisek a definovaný posuvník s rozmezím -15 až 15 dílků nastaven na hodnotu 1

(11) určíme, co bude daný posuvník provádět za změnu. V okně grafů se bude zobrazovat implicitně zadané rovnice, první se načte z textového pole TF1 a druhá (12) se vytvoří načtením hodnot z jednotlivých posuvníků, jež doplní

danou rovnici roviny. Parametr onchange udává, co se provede při změně na posuvníku. (13) Ten bude jako akci provádět změnu v okně grafů PL1 a to takovou, že znovu vypíše implicitně zadané rovnice (jednu načte z TF1 a (14) druhou si poskládá z hodnot na posuvníku). Určíme rozmezí na osách x, y, z, které se nám zobrazí. (15) Popisky os budou x, y, z a nastavíme rovnoměrné zobrazování na osách. Určíme počet vykreslovaných bodů. (16) Barva plochy bude modrá a barva roviny bude červená. Osy se nám zobrazí jako rámy okolo zobrazovaných objektů. Tím jsme ukončili definování akcí pro posuvník.

(17) zobrazí se nám zobáček na posuvníku a posuvník se rozdělí na větší celky po pěti a (18) nejmenší dílek bude po jedničkách.

(19) – (27) stejné jako viz výše (posuvník)

(28) – (36) stejné jako viz výše (posuvník)

(37) – (45) stejné jako viz výše (posuvník)

(46) definujeme okno grafů, ve kterém budeme zobrazovat implicitně zadané rovnice, ale při prvním zobrazení uvidíme jen osy. Určíme rozmezí na osách x, y, z, (47) které se nám zobrazí, popisky os, rozvržení os a (48) počet vykreslených bodů.

(49) textové pole, do kterého nelze nic zapisovat a zobrazuje se do jedné řádky po 30 znacích.

(50) tlačítko označené jako B5 se jmenuje ***Vykresli*** a (51) bude nám po stisknutí zobrazovat implicitně zadané rovnice z textového pole TF1 a (52) z jednotlivých posuvníků poskládá rovnici roviny. Nastaví osy x, y, z, (53) počet vykreslovaných bodů, barvu plochy a roviny a typ zobrazení os.

(54) tlačítko ***Vypis*** nám bude po stisknutí vypisovat do textového pole TB1 rovnici (55) roviny.

(56) tlačítko ***Vyčistit***, bude (57) po stisknutí měnit text v textovém poli TF1(58) a TB1(59) na prázdný řetězec. (60) V okně grafů nám zobrazí počátek souřadnic přes zobrazení funkce (61) s nastavenými osami.(62)

(63) tlačítko ***OK*** nám bude zavírat celé okno aplikace a jako výsledek zapíše do Maple nastavené hodnoty v textovém poli TF1 a v jednotlivých posuvnících.

(64) tlačítko ***Napoveda*** nám vyvolá dialogové okno se zprávou MD1

(65) – (67) ukončovácí závorky k uspořádání prvků v mapletu a k uzavření funkce *Window()*, kde jsou definovány všechny prvky mapletu

(68) zpráva pro zobrazení v nápovědě označená jako MD1 s textem nápovědy

(69) – (73) text nápovědy, znakem \n se odřádkuje

(74) text nápovědy bude jen informativní (v okně nápovědy se zobrazí kromě textu nápovědy i obrázek s I)

(75) ukončení mapletu

(76) zobrazení mapletu s názvem maplet

5. Využití Maple ve výuce analytické geometrie

5.1. Přehled vytvořených pomůcek

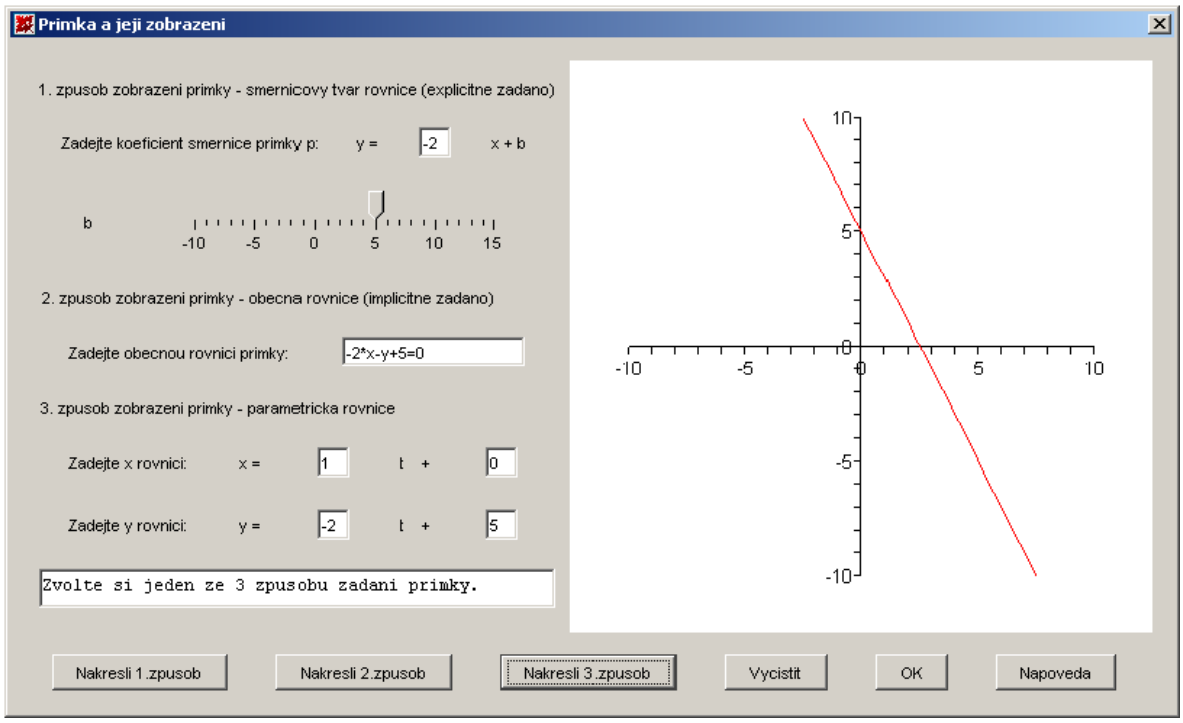
Seznam mapletů připravených pro využití ve výuce:

1. Přímka a její vyjádření
2. Vzájemná poloha dvou přímek
3. Kružnice a její zobrazení
4. Elipsa a její zobrazení
5. Hyperbola a její zobrazení
6. Parabola a její zobrazení
7. Vzájemná poloha kuželosečky a přímky
8. Vzájemná poloha dvou kuželoseček
9. Rovina a její vyjádření
10. Vzájemná poloha dvou rovin
11. Plocha a její zobrazení
12. Vzájemné polohy plochy a roviny
13. Vzájemná poloha dvou ploch

Jednotlivé maplety jsou navrženy tak, by jejich obsluha byla co nejjednodušší a jejich ovládání okamžitě každý pochopil. Zároveň jsem se snažila připravit uživateli co největší počet různých polí a posuvníků, aby si sám mohl libovolně měnit parametry a sledovat změny v zobrazení rovnic a zjistit, který parametr má vliv na jakou vlastnost, a celkově více vizualizovat rovnice.

5.2. Popis vytvořených pomůcek

5.2.1. Parametrické a obecné vyjádření přímky



obr. 5.1 – Přímka a její zobrazení (maplet)

Maplet ukazuje tři způsoby zapsání rovnice přímky.

- První způsob zapsání přímky je ve směrnicovém tvaru $y=ax+b$, kde a je směrnice přímky, b je posunutí přímky po ose y. Směrnice přímky nám udává sklon přímky. Dá se vypočítat pomocí úhlu, a to $a=tg(\varphi)$, kde φ je úhel, který svírá přímka se směrem kladné poloosy x. Posunutí po ose y nám udává koeficient b. Můžeme si vyzkoušet posunutím „zobáčku“ po posuvníku.

- Druhým způsobem zapsání přímky je obecná rovnice $ax+by+c=0$. Takovému způsobu říkáme, že je zapsán explicitně. Do obecné rovnice se vždy zapisuje normálový vektor. Tento vektor (a, b) je kolmý na směrový vektor přímky. Z našeho příkladu můžeme vyčíst, že normálový vektor bude $\vec{n}=(-2, -1)$.

- Třetím způsobem zadání přímky je zadání parametrickou rovnicí. K tomu budeme potřebovat bod (př. A[x,y]) a vektor (př. $\vec{u}=(u,v)$), případně dva body (př. K[k_1,k_2], L[l_1,l_2]). Parametrická rovnice je ve tvaru $X=A+\vec{u}\cdot t$, popř. $X=K+\overrightarrow{KL}\cdot t$, kde KL je vektor z bodu K do bodu L. Skládá se ze dvou rovnic a to x-ové rovnice a y-ové rovnice. Z našeho příkladu můžeme z parametrické rovnice vyčíst, že přímka bude procházet bodem M[0,5] a směrový vektor bude $\vec{s}=(1,2)$.

Pro konkrétní počítání v programu Maple budeme potřebovat otevřít knihovnu **Geometry** příkazem *with(geometry)*. Z této knihovny může použít následující příkazy: *point(P, Px, Py)*, *point(P, [Px, Py])*.

```
> with(geometry):
  point(A, -1, 2), point(B, [3, -2]);
```

A, B

V této knihovně je několik obecných funkcí, pomocí kterých můžeme zjišťovat další podrobnosti daného objektu.

Příkazem *form(P)* zjistíme jakým druhem objektu je P (např. Bod, přímka, elipsa atd.).

```
> form(A);
```

point2d

Příkaz *coordinates(P)* nám vrátí souřadnice bodu P (např. [1,1]).

```
> coordinates(A), coordinates(B);
```

[-1, 2], [3, -2]

HorizontalCoord(P) nám vrátí x-ovou souřadnici bodu P.

VerticalCoord(P) vrátí y-ovou souřadnici bodu P.

```
> HorizontalCoord(A);
  VerticalCoord(A);
```

-1

2

Obecným příkazem *detail(P)* získáme informace o bodu P (např. druh objektu, souřadnice atd.).

```
> detail(B);
```

name of the object: B

form of the object: point2d

coordinates of the point: [3, -2]

Dalším vhodným příkazem bude příkaz pro vytvoření přímky. Přímku můžeme zadat dvěmi body příkazem *line(l, [A, B])* nebo rovnicí a to pomocí příkazu line(l, rovnice, seznam dvou jmen reprezentujících jména os).

```
> line(p, x-3*y, [x,y]);
```

$$p$$

Další příkaz, který by se nám mohl hodit je *distance(P, l)*, kde P je definovaný bod a l je definovaná přímka a příkaz vypočítá vzdálenost bodu od přímky.

```
> distance(A, p);
```

$$\frac{7}{10}\sqrt{10}$$

Příkaz midpoint(název, bod A, bod B) nám vrátí souřadnice nového body s určeným názvem, který se nachází v prostředku úsečky AB.

```
> point(A, 0, 0), point(B, 2, 0); midpoint(S, A, B);
  coordinates(S);
```

$$A, B$$

$$S$$

$$[1, 0]$$

Příkazem ParallelLine(název, bod, přímka) získáme rovnoběžnou přímku s danou přímkou, procházející daným bodem.

```
> point(M, 2, 3), line(p, x+y=1, [x,y]);
  ParallelLine(q, M, p); Equation(q);
```

$$M, p$$

$$q$$

$$-5 + x + y = 0$$

Příklad:

Zapište rovnici přímky q, která bude procházet body A[-2, 3] a B[4, -1].

Možné řešení pomocí programu Maple:

```
> restart:with(geometry):
  _EnvHorizontalName := x: _EnvVerticalName := y:
  point(A, [-2, 3]); point(B, 4, -1);
```

$$A$$

$$B$$

```
> coordinates(A); coordinates(B);
```

$$[-2, 3]$$

$$[4, -1]$$

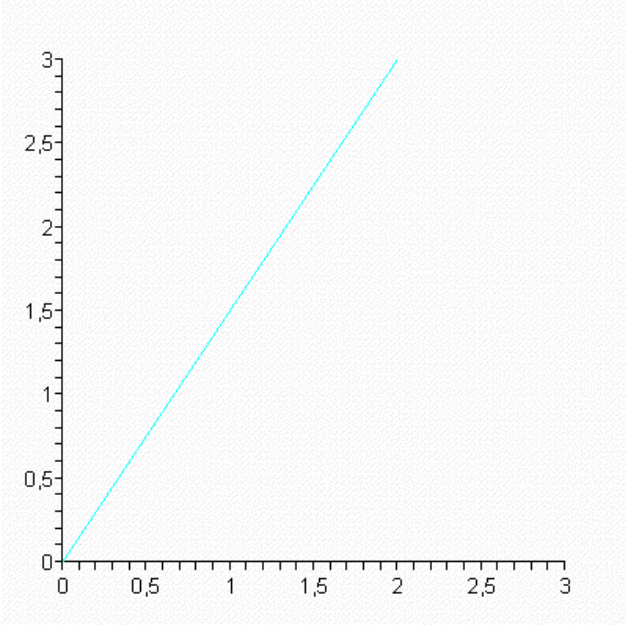
```
> line(l, [A, B]);
```

$$l$$

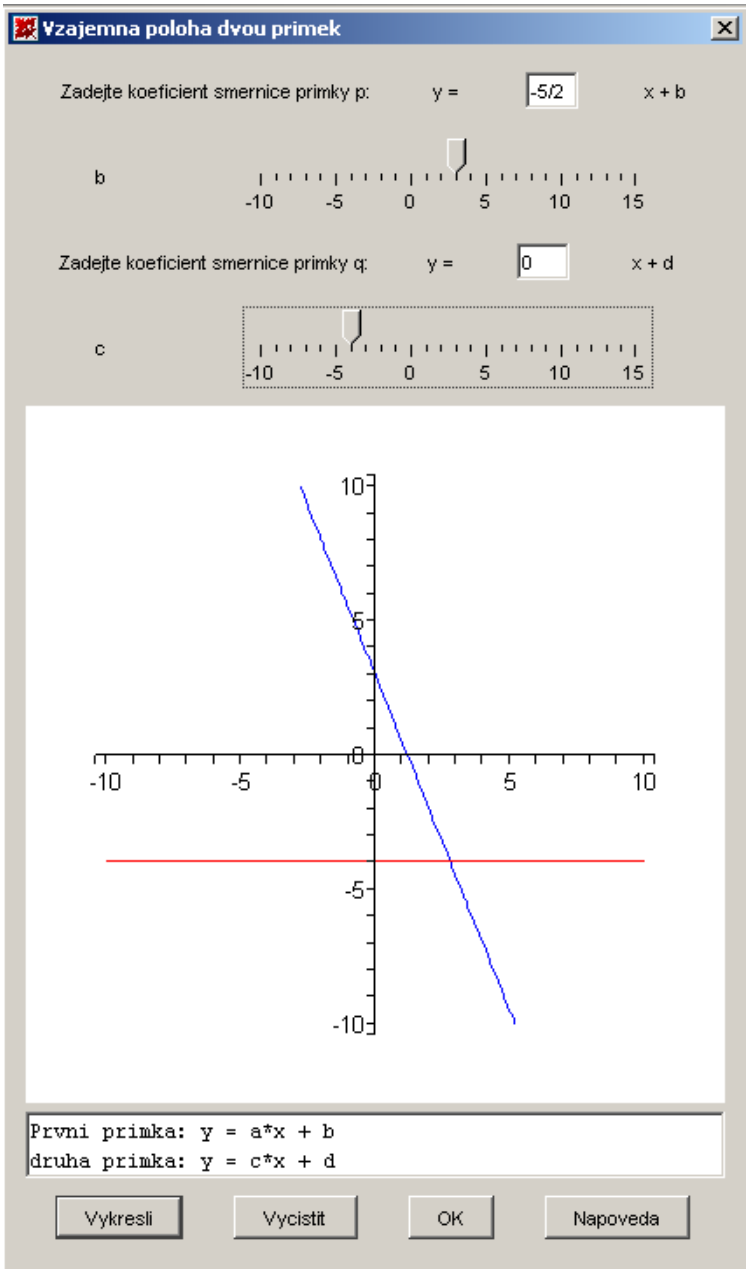
```
> Equation(l);
```

$$-10 + 4\,x + 6\,y = 0$$

```
> with(plots):
  PLOT(CURVES([[0,0],[2,3]]),COLOR(HUE,0.5),VIEW(-0..3,0..3));
```



5.2.2. Vzájemná poloha dvou přímek



obr. 5.2 – Vzájemná poloha dvou přímek (maplet)

Daný maplet zobrazuje vzájemnou polohu dvou přímek zadaných ve směrnicovém tvaru. Každá přímka je zadána směrnicí a posunutím přímky. Zadaná směrnice v textovém poli se následně může měnit. Pro její opětovné zobrazení je nutno znovu stisknout tlačítko ***Vykresli***. Pro změnu parametru posunutí to neplatí. Při změně hodnoty se na posuvníku změní vykreslená přímka automaticky.

Maplet ukazuje typ úlohy na určení vzájemné polohy dvou přímek. Dané polohy mohou být: rovnoběžné, totožné či různoběžné. Vzájemnou polohu určuje lineární závislost či nezávislost směrových $\left(\overrightarrow{s_q} = k \cdot \overrightarrow{s_p}\right)$ či normálových vektorů $\left(\overrightarrow{n_q} = k \cdot \overrightarrow{n_p}\right)$ přímek. Dále budu v tomto textu uvádět směrové vektory.

Pokud jsou vektory lineárně závislé $\overrightarrow{s_q} = k \cdot \overrightarrow{s_p}$, může být vzájemná poloha přímek rovnoběžná nebo totožná. O jakou polohu půjde, zjistíme s použitím libovolného bodu, který náleží jedné z přímek. Pokud náleží i druhé přímce, budou přímky totožné. Pokud druhé přímce náležet nebudou, budou přímky rovnoběžné, nikoli totožné.

V případě, že vektory přímek budou lineárně nezávislé $\overrightarrow{s_q} \neq k \cdot \overrightarrow{s_p}$, bude se jednat o vzájemnou polohu různoběžnou. Z toho plyne, že přímky budou mít jeden společný bod – průsečík.

Při výpočtu průsečíků dvou přímek se vychází ze zadaných rovnic přímek

$$y = ax + b$$
$$y = cx + d$$

Pokud bude platit rovnost $a = c$, je jasné, že rovnice mají stejnou směrnici a tudíž jsou rovnoběžné.

Pokud by platily rovnosti $a = c, b = d$, budou mít nejen stejnou směrnici, ale i posunutí. Z toho vyplývá, že to budou dvě totožné přímky.

Pro případ, kdy $a \neq c$ budou mít přímky různou směrnici, budu tedy různoběžné a můžeme spočítat jejich průsečík.

Po dosazení jedné rovnice do druhé a úpravě dostáváme x hodnotu průsečíku

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

Po dosazení hodnoty x do jedné z původní zadaný rovnice dostaneme hodnotu pro y.

$$y = \frac{ad - bc}{a - c}$$

Vypočítali jsem tedy x a y souřadnici průsečíku společného pro obě zadané přímky.

Příklad:

Přímka p má rovnici $2x + 3y = 0$, přímka q prochází bodem M[-1,7] a je rovnoběžná s přímkou $3x + y - 4 = 0$. Určete vzájemnou polohu přímek p a q.

Možné řešení pomocí programu Maple:

```
> pp:=Equation(line(p, 2*x+3*y, [x,y]));
```

$$pp := 2x + 3y = 0$$

```
> point(M, -1, 7); line(l, 3*x+y=4, [x,y]);
  ParallelLine(q, M, l);
```

$$M$$

$$l$$

$$q$$

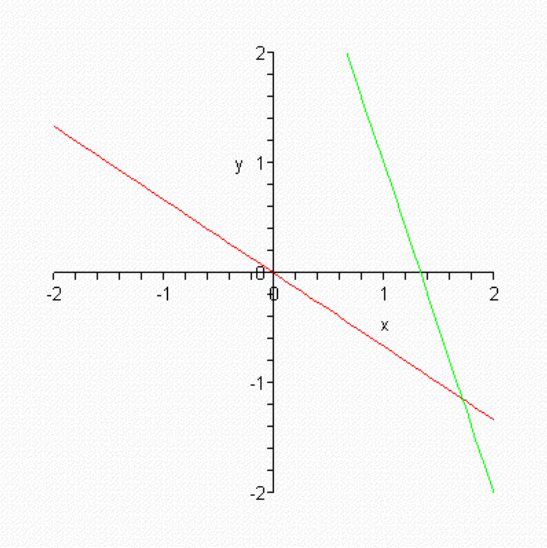
```
> qq:=Equation(q);
```

$$qq := 3x + y - 4 = 0$$

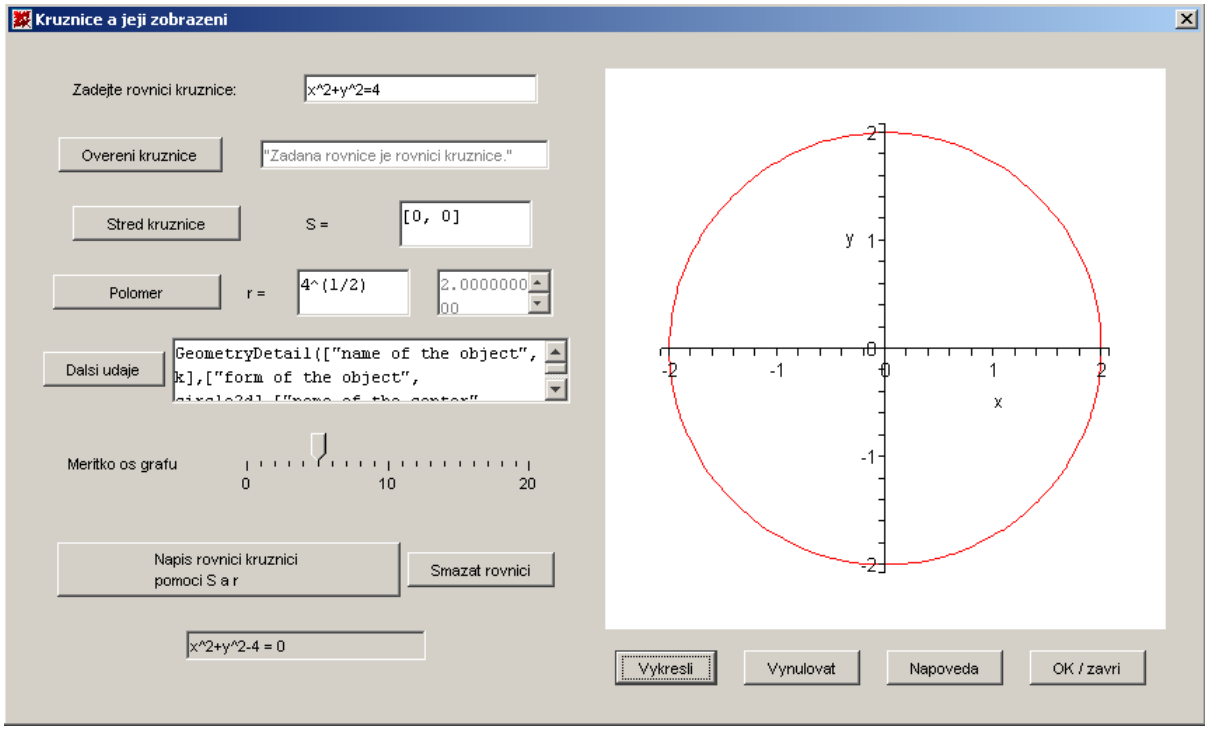
```
> solve({pp,qq}, {x,y});
```

$$\left\{ y = \frac{-8}{7}, x = \frac{12}{7} \right\}$$

```
> with(plots):
  plot([-2/3*x, 4-3*x], x=-2..2, y=-2..2);
```



5.2.3. Kružnice a její zobrazení



obr. 5.3 – Kružnice a její zobrazení (maplet)

Maplet zobrazuje kružnici ze zadané rovnice. Pro vykreslení kružnice tlačítkem ***Vykresli*** budeme potřebovat nejdříve ověřit, zda zadaná rovnice je opravdu rovnicí kružnice. To uděláme pomocí tlačítka ***Overeni kruznice.*** Stiskem tlačítka ***Střed kružnice*** zjistíme střed této kružnice. Tlačítkem ***Polomer*** zjistíme poloměr této kružnice. Tlačítko ***Vykresli*** nám zobrazí zadanou rovnici kružnice. Budeme-li chtít zobrazit kružnici zadanou pomocí středu a poloměru kružnice, můžeme tyto hodnoty nastavit v textovém poli pro tyto hodnoty. Pro samotné zobrazení pak stiskneme tlačítko ***Napis rovnici kruznice pomocí S a r***, které nám zapíše příslušnou rovnici kružnice vypočítanou ze středu a poloměru. Tlačítko ***Vynulovat*** smaže všechny hodnoty a vrátí maplet do původního stavu jako po jeho spuštění. Stiskem tlačítka ***Napoveda*** se nám zobrazí nápověda k ovládání mapletu.

Středová rovnice kružnice je ve tvaru $(x-m)^2+(y-n)^2=r^2$, kde m,n jsou souřadnice středu a r je poloměr kružnice. Kromě tvaru středové rovnice

kružnice máme ještě obecný tvar rovnice. Ten má tvar $x^2 + y^2 + mx + ny + p = 0$, kde musí platit $m^2 + n^2 > 4p$.

Pro konkrétní počítání v programu Maple budeme opět potřebovat otevřít knihovnu **Geometry** příkazem *with(geometry)*. Příkaz pro vytvoření kružnice je *circle()*. Zápis je možný pomocí tří bodů, pomocí bodu a poloměru, pomocí rovnice. Samozřejmě tyto body, poloměry a rovnice musíme mít předem definované. Circle(název, jeden ze tří možných postupů zápisu, seznam dvou jmen reprezentujících názvy os). Můžeme připsat název středu kružnice pomocí parametru *'centername'*.

```
> with(geometry):
  _EnvHorizontalName := x: _EnvVerticalName := y:
```

Kružnice zadaná třemi body

```
> circle(k1,[point(A,0,0), point(B,2,0),
  point(C,1,2)],'centername'=O1):Equation(k1);
```

$$x^2 + y^2 - 2x - \frac{3}{2}y = 0$$

Kružnice zadaná středem a poloměrem

```
> circle(k2,[point(S,2,1),3]): Equation(k2);
```

$$-4 + x^2 + y^2 - 4x - 2y = 0$$

Kružnice zadaná rovnicí

```
> circle(k3, x^2+y^2=4): Equation(k3);
```

$$x^2 + y^2 - 4 = 0$$

Pro kružnici můžeme zjišťovat tyto hodnoty:

rovnice kružnice - *Equation(k);*

```
> circle(k3, x^2+y^2=4): Equation(k3);
```

$$x^2 + y^2 - 4 = 0$$

střed kružnice - *center(k);* pro získání souřadnic tohoto středu zadáme příkaz

```
> coordinates(center(k3));
```

$$[0, 0]$$

poloměr kružnice – *radius(k);*

```
> radius(k3);
```

$$\sqrt{4}$$

```
> simplify(radius(k3));
```

$$2$$

Jednotlivé souřadnice středu kružnice získáme pomocí *HorizontalCoord(S)* a *VerticalCoord(S).*

```
> HorizontalCoord(S); VerticalCoord(S);
```

$$2$$

$$1$$

Příklad:

Zjistěte rovnici kružnice k, která má obsah 78,54 a střed v bodě S[5,6]. Určete průsečíky kružnice k s osou x a s osou y.

Možné řešení pomocí programu Maple:

```
> polomer:= sqrt(78.54/Pi);
  simplify(polomer);
```

$$polomer := 8.862279616 \sqrt{\frac{1}{\pi}}$$

$$5.000005845$$

```
> circle(k,[point(S,5,6),polomer]):
  Equation(k);
```

$$x^2 + y^2 - 10\,x - 12\,y + 61 - \frac{78.53999999}{\pi} = 0$$

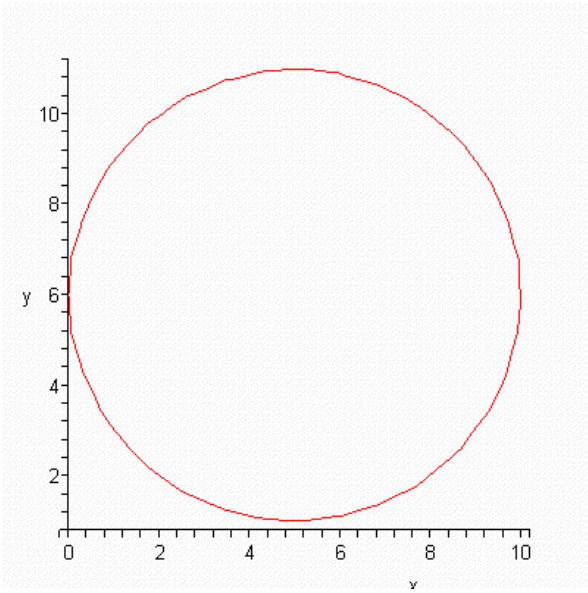
```
> solve({x^2+y^2-10*x-12*y+61-78.53999999/Pi = 0,x=0});
```

$$\{x = 0.,\ y = 5.992354237\},\ \{x = 0.,\ y = 6.007645763\}$$

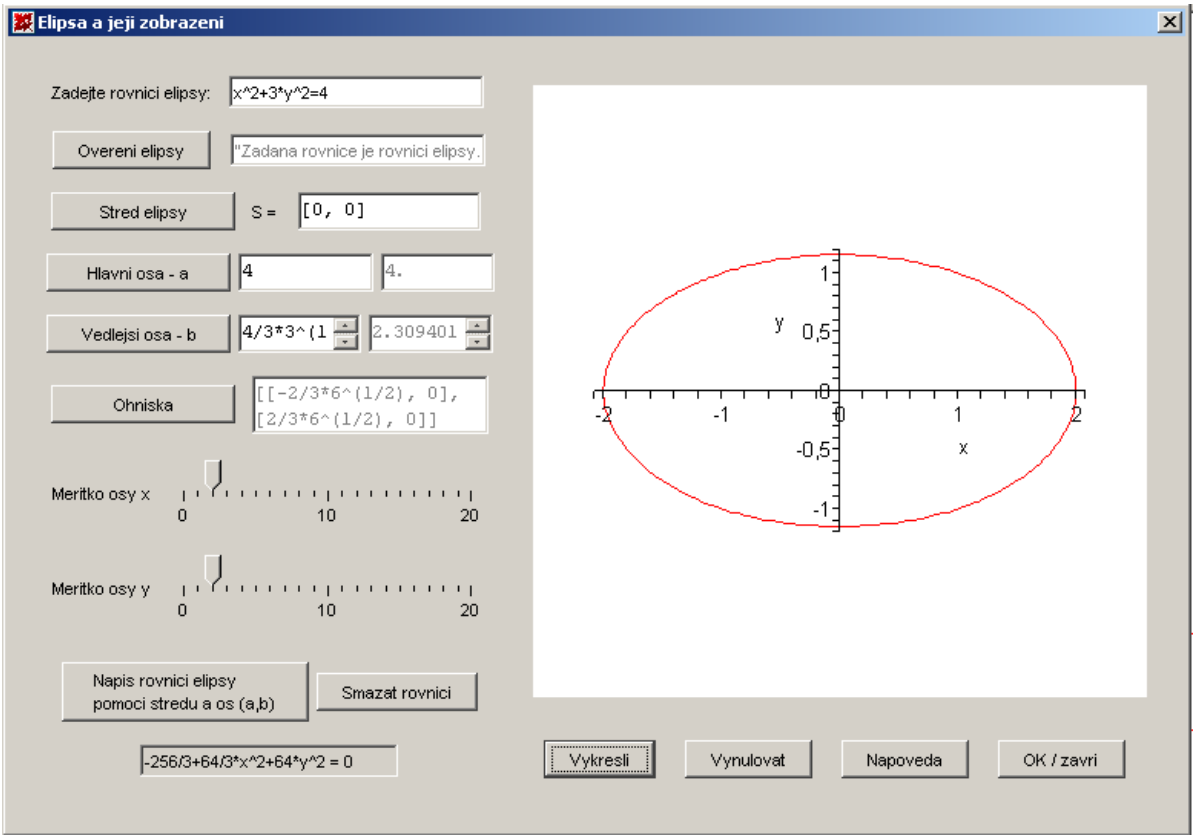
```
> solve({x^2+y^2-10*x-12*y+61-78.53999999/Pi = 0,y=0});
```

$$\{x = 5.000000000 - 3.316615978\,I,\ y = 0.\},\ \{x = 5. + 3.316615978\,I,\ y = 0.\}$$

```
> with(plots):
   implicitplot(x^2+y^2-10*x-12*y+61-78.53999999/Pi = 0,
   x=-10..11, y=-10..11);
```


10
8
y 6
4
2
0
2
4
6
8
10
x

5.2.4. Elipsa a její zobrazení



obr. 5.4 – Elipsa a její zobrazení (maplet)

Maplet je podobný mapletu Kružnice. Po zadání rovnice ověříme, zda se opravdu jedná o elipsu pomocí tlačítka ***Overeni elipsy***. Tím se nám zpřístupní tlačítko ***Vykresli*** pro zobrazení elipsy. Podrobnosti o elipse zjistíme pomocí tlačítek ***Stred elipsy, Hlavni osa, Vedlejsi osa, Ohniska***.

Středová rovnice elipsy je ve tvaru $\frac{(x-m)^2}{a^2}+\frac{(y-n)^2}{b^2}=1$, kde m,n jsou souřadnice středu, a je velikost hlavní osy elipsy, b je velikost vedlejší osy elipsy. Bude-li $a=b$, bude to speciální případ elipsy, neboť obě osy budou stejně dlouhé a bude se jednat o kružnici s poloměrem a. Kromě tvaru středové rovnice elipsy máme ještě rovnici obecného tvar $kx^2+ly^2+mx+ny+p=0$, kde musí platit $k\neq l, k>0, l>0$ a dále $\frac{m^2}{4k}+\frac{n^2}{4l}-p>0$.

Po otevření knihovny **Geometry** příkazem *with(geometry)*, můžeme definovat elipsu pomocí rovnice, řídící přímky, excentricity, ohniska, vzdáleností mezi vrcholy. Příklad ellipse(název elipsy, jeden ze způsobů zobrazení, seznam jmen os).

Elipsa zadaná rovnicí

```
> ellipse(e1, 2*x^2+y^2-4*x+4*y=0): Equation(e1);
```

$$2x^2 + y^2 - 4x + 4y = 0$$

Elipsa zadaná řídící přímkou, ohniskem F a excentricitou e

```
> line(l,x=-2,[x,y]): point(f,1,0): e := 1/2:
  ellipse(e2,['directrix'=l,'focus'=f,'eccentricity'=e]):
  Equation(e2);
```

$$\frac{3}{4}x^2 - 3x + y^2 = 0$$

Elipsa zadaná vzdáleností mezi vrcholy A, B a ohnisky E, F

```
> point(A,4,0), point(B,-4,0), point(E,0,3), point(F,0,-3):
  ellipse(e4,['MajorAxis'=[A,B],'MinorAxis'=[E,F]]):
  Equation(e4);
```

$$144x^2 - 2304 + 256y^2 = 0$$

Elipsa zadaná řídící přímkou, ohniskem F a excentricitou e

```
> ellipse(e1, 2*x^2+y^2-4*x+4*y=0):
  foci(e2), map(coordinates,foci(e2));
```

$$[foci\_1\_e2, foci\_2\_e2], [[1, -2 - \sqrt{3}], [1, -2 + \sqrt{3}]]$$

```
> MajorAxis(e2);
```

$$2\sqrt{6}$$

```
> ellipse(e5,['foci'=foci(e2),'MajorAxis'=MajorAxis(e2)]):
  Equation(e5);
```

$$-192x + 192y + 96x^2 + 48y^2 = 0$$

Další z příkazů otevřené knihovny vybereme tyto příkazy:

center(e), který nám zobrazí střed elipsy,

```
> ellipse(e,2*x^2+y^2-4*x+4*y=0):
  center(e), coordinates(center(e));
```

$$center\_e, [1, -2]$$

Equation(e) nám vrátí rovnici elipsy,

```
> ellipse(e,2*x^2+y^2-4*x+4*y=0): Equation(e);
```

$$2x^2 + y^2 - 4x + 4y = 0$$

Foci(e) nám zobrazí ohniska elipsy,

```
> foci(e), map(coordinates,foci(e));
```

$$[foci\_1\_e, foci\_2\_e], [[1, -2 - \sqrt{3}], [1, -2 + \sqrt{3}]]$$

MajorAxis(e) je velikost hlavní osy elipsy od středu elipsy,

```
> MajorAxis(e);
```

$$2\sqrt{6}$$

MinorAxis(e) je velikost vedlejší osy elipsy od jejího středu,

```
> MinorAxis(e);
```

$$2\sqrt{3}$$

form(e) nám ukáže jakého druhu kuželosečky je daná rovnice,

```
> form(e);
```

ellipse2d

detail(e) nám zobrazí všechny podrobnosti od dané elipse (kuželosečce).

```
> detail(e);
```

name of the object: e

form of the object: ellipse2d

center: [1, -2]

foci: [[1, -2-3^(1/2)], [1, -2+3^(1/2)]]

*length of the major axis: 2*6^(1/2)*

*length of the minor axis: 2*3^(1/2)*

*equation of the ellipse: 2*x^2+y^2-4*x+4*y = 0*

Příklad:

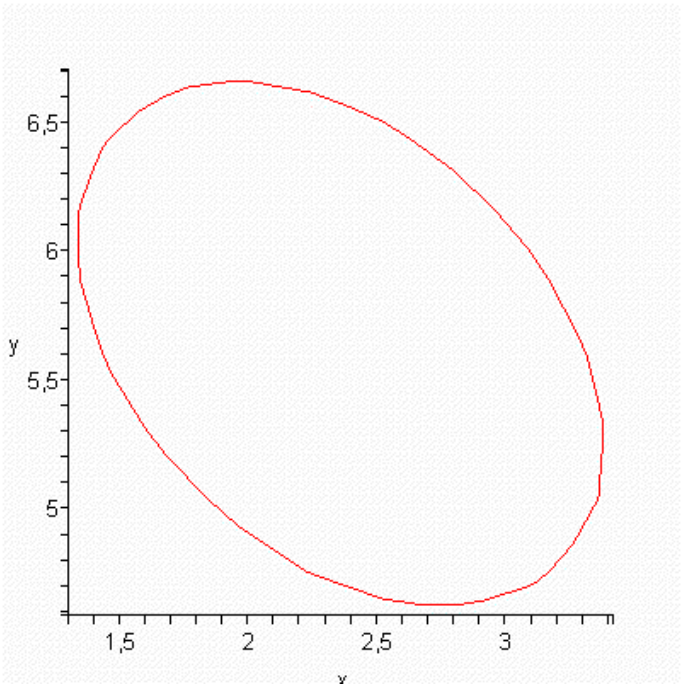
Zjistěte rovnici elipsy, která má řídící přímku y=x+1, ohnisko v bodě $E[3,5]$ a excentricitu ¾.

Možné řešení pomocí programu Maple:

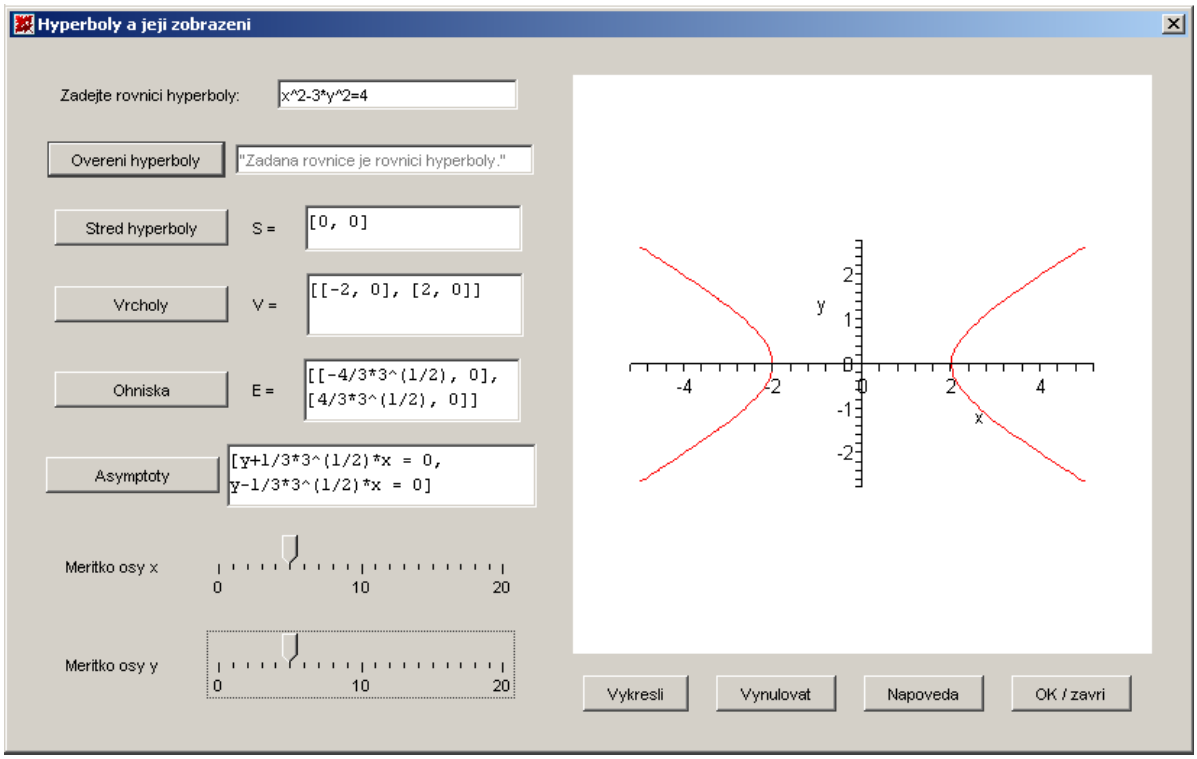
```
> restart:  with(geometry):
  _EnvHorizontalName := x: _EnvVerticalName := y:
> line(l,y=x+1,[x,y]): point(f,3,5): e := 3/4:
  ellipse(eq,['directrix'=l,'focus'=f,'eccentricity'=e]):
  Equation(eq);
```

$$\frac{1079}{16} + \frac{23}{16}x^2 + \frac{9}{8}xy + \frac{23}{16}y^2 - \frac{105}{8}x - \frac{151}{8}y = 0$$

```
> with(plots):
implicitplot(1079/16+23/16*x^2+9/8*x*y+23/16*y^2-105/8*x-
151/8*y = 0, x=0..7, y=0..7);
```



5.2.5. Hyperbola a její zobrazení



obr. 5.5 – Hyperbola a její zobrazení (maplet)

Maplet zobrazuje hyperbolu pomocí obecné rovnice. Po ověření tlačítkem ***Overeni hyperboly***, zda zadaná rovnice jej opravu rovnice hyperboly, můžeme zjistit střed hyperboly, její vrcholy, ohniska a asymptoty. Vše po stisknutí příslušného tlačítka. Posuvníkem ovládáme měřítka os a to symetricky na obě strany osy o stejnou hodnotu.

Středová rovnice hyperboly je ve tvaru $\frac{(x-m)^2}{a^2}-\frac{(y-n)^2}{b^2}=1$, kde m,n jsou souřadnice středu, a je velikost hlavní osy hyperboly, b je velikost vedlejší osy. Bude-li $a=b$, bude to speciální případ hyperboly, neboť bude rovnoosá. Kromě tvaru středové rovnice hyperboly máme ještě rovnici obecného tvaru $kx^2+ly^2+mx+ny+p=0$, kde musí platit $k\cdot l<0$ a dále $\frac{m^2}{4k}+\frac{n^2}{4l}-p\neq 0$.

V programu Maple určíme hyperbolu několika způsoby. Můžeme ji zadat pěti body nebo pomocí řídící přímky, ohniska a excentricity nebo seznamy dvou ohnisek a dvou vrcholů nebo seznamem dvou bodů vrcholů a vzdáleností ohnisek nebo seznamem dvou ohnisek a vzdáleností vrcholů nebo pomocí rovnice: hyperbola (název hyperboly, jeden ze způsobů zobrazení, seznam jmen os).

Hyperbola zadaná pěti body

```
> with(geometry):
  point(A,2,0): point(B,-2,0): point(C,4,9): point(E,4,-9):
  point(F,3,6):
  hyperbola(h1, [A, B, C, E,F], [x,y]); Equation(h1);
```

$$h1$$

$$-139968 + 7776\,y - 1944\,x\,y + 34992\,x^2 - 5184\,y^2 = 0$$

Hyperbola řídící přímkou, ohniskem a excentricitou

```
> line(l,x=-2,[x,y]): point(f,1,0): e := 3/2:
hyperbola(h2,['directrix'=l,'focus'=f,'eccentricity'=e],[x,y])
:
eq := Equation(h2);
```

$$eq := -\frac{5}{4}x^2 - 11\,x - 8 + y^2 = 0$$

Hyperbola zadaná rovnicí

```
> hyperbola(h3,9*y^2-4*x^2=36,[x,y]):Equation(h3);
```

$$9\,y^2 - 4\,x^2 - 36 = 0$$

Podrobnosti o hyperbole zjistíme pomocí příkazů:

center(h), který nám zobrazí střed hyperboly,

```
> coordinates(center(h3));
```

$$[0, 0]$$

foci(h) vypíše ohniska hyperboly,

```
> foci(h3): map(coordinates,foci(h3));
```

$$[[0, -\sqrt{13}], [0, \sqrt{13}]]$$

vertices(h) vypíše vrcholy hyperboly,

```
> vertices(h3): map(coordinates,vertices(h3));
```

[[0, -2], [0, 2]]

asymptotes(h) vrátí rovnice asymptot hyperboly,

```
> asymptotes(h3), map(Equation,asymptotes(h3));
```

$$\left[\frac{2}{3}x+y=0,\ -\frac{2}{3}x+y=0\right]$$

Equation(h) vrátí rovnici hyperboly.

```
> Equation(h3);
```

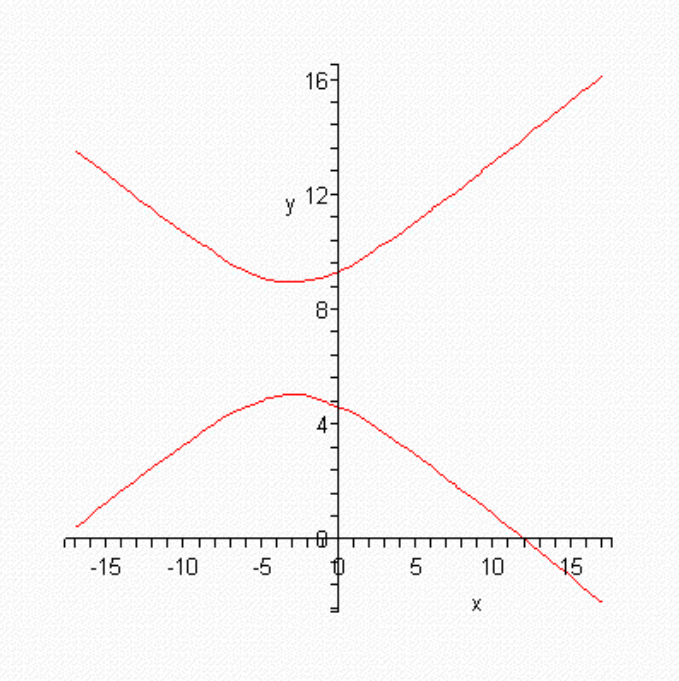
$$-36-4x^2+9y^2=0$$

Příklad:

Zjistěte rovnici hyperboly, která má vrcholy stejné jako hyperbola $\frac{(y-7)^2}{4}-\frac{(x+3)^2}{2}=1$ a má dvojnásobnou vzdálenost mezi ohnisky.

Možné řešení pomocí programu Maple:

```
> with(plots):
  implicitplot(-13824+384*x+4480*y+64*x^2-320*y^2 = 0,
  x=-17..17, y=-17..17);
```



```
> hyperbola(hh,2*(y-7)^2-4*(x+3)^2=8,[x,y]);
  vertices(hh): map(coordinates,vertices(hh));
  foci(hh): map(coordinates,foci(hh));
```

$$hh$$

$$[[-3, 5], [-3, 9]]$$

$$[[-3, 7 - \sqrt{6}], [-3, 7 + \sqrt{6}]]$$

```
> hyperbola(h,['vertices'=vertices(hh),'distancef'=2*distance(
  op(foci(hh)))],[x,y]):
  vertices(h): map(coordinates,vertices(h));
  foci(h): map(coordinates,foci(h));
```

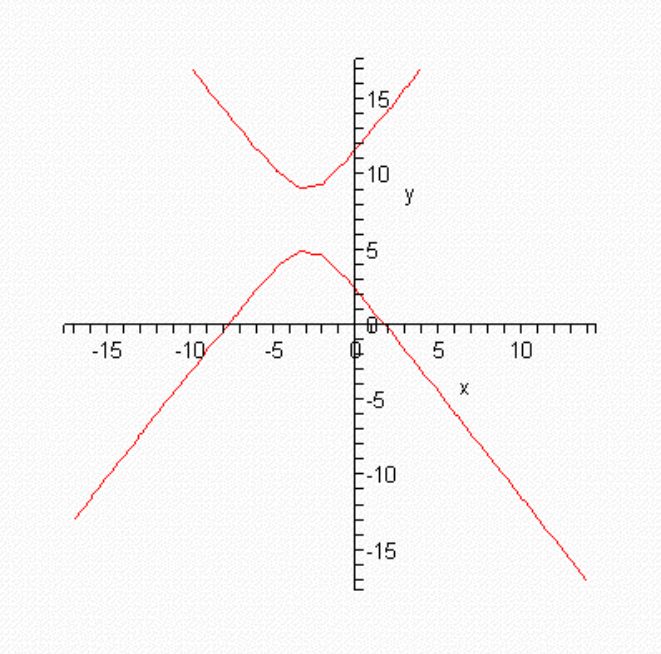
$$[[-3, 5], [-3, 9]]$$

$$[[-3, 7 - 2\sqrt{6}], [-3, 7 + 2\sqrt{6}]]$$

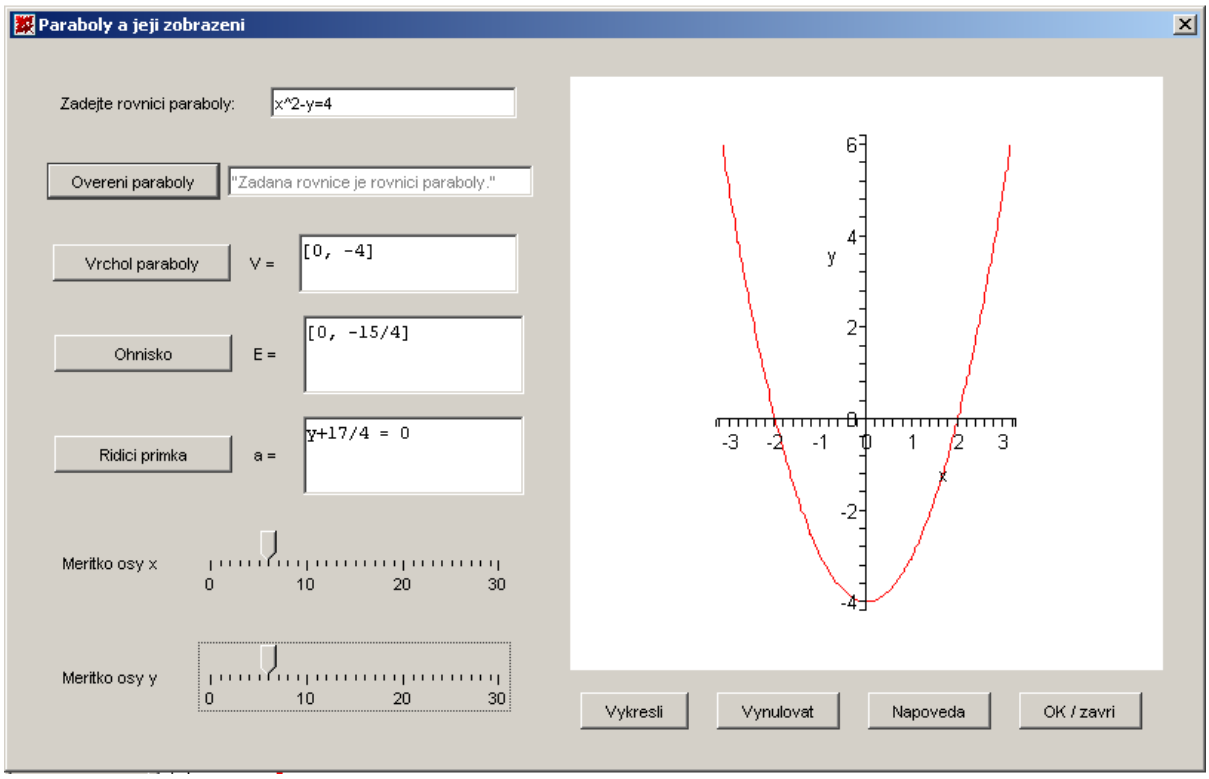
```
> Equation(h);
```

$$-13824 + 384\,x + 4480\,y + 64\,x^2 - 320\,y^2 = 0$$

```
> with(plots):
  implicitplot(2*(y-7)^2-4*(x+3)^2=8, x=-17..17, y=-17..17);
```



5.2.6. Parabola a její zobrazení



obr. 5.6 – Parabola a její zobrazení

Maplet je podobný předchozím mapletům. Po zjištění, že je to opravdu rovnice paraboly se nám zpřístupní tlačítko ***Vykresli***. Dále můžeme zjistit vrchol paraboly, ohnisko a řídící přímku paraboly. Opět můžeme nastavovat měřítko na osách.

Středová rovnice paraboly je ve tvaru $(y-n)^2 = \pm 2p(x-m)$ nebo $(x-n)^2 = \pm 2p(y-m)$, kde m,n jsou souřadnice středu, p je vzdálenost ohniska od řídící přímky. Kromě tvaru středové rovnice paraboly máme ještě rovnici obecného tvaru $ly^2 + mx + ny + p = 0$, kde musí platit $l \cdot m \neq 0$. Parabola pak bude mít osu paraboly o || x. Nebo parabola může mít tvar $kx^2 + mx + ny + p = 0$, kde musí platit $k \cdot n \neq 0$ a parabola bude mít osu paraboly o || y.

Po otevření knihovny **Geometry**, můžeme definovat parabolu pomocí pěti bodů nebo ohniskem a vrcholem paraboly nebo řídící přímkou a ohniskem nebo rovnicí. Obecný zápis příkazu pro vytvoření paraboly bude vypadaná následovně parabola (název paraboly, jeden ze způsobů zobrazení, seznam jmen os).

Parabola zadaná pěti body

```
> point(A,-6,3+4*sqrt(3)),point(B,-5,9),
  point(C,-4,3+2*sqrt(6)),point(E,-3,3+2*sqrt(3)),
  point(F,-2,3):
  parabola(p1,[A,B,C,E,F],[x,y]):
  Equation(p1);
```

$$(-336\sqrt{3} + 144\sqrt{3}\sqrt{2} - 144\sqrt{2} + 432)y^2 + (5184 + 1728\sqrt{3}\sqrt{2} - 1728\sqrt{2} - 4032\sqrt{3})x$$

$$+ (-2592 - 864\sqrt{3}\sqrt{2} + 864\sqrt{2} + 2016\sqrt{3})y + 14256 + 4752\sqrt{3}\sqrt{2} - 4752\sqrt{2} - 11088\sqrt{3} = 0$$

Parabola zadaná řídící přímkou a ohniskem

```
> line(p, x=-2, [x,y]): point(f,1,0):
  parabola(p2, ['directrix'=p, 'focus'=f], [x,y]):
  Equation(p2);
```

$$-6x - 3 + y^2 = 0$$

Parabola zadaná ohniskem a vrcholem

```
> parabola(p3, ['focus'=point(f,1,0),
  'vertex'=point(V,0,0)], [x,y]):
  Equation(p3);
```

$$-4x + y^2 = 0$$

Parabola zadaná rovnicí

```
> parabola(p4,y^2+12*x-6*y+33=0):
  Equation(p4);
```

$$y^2 + 12x - 6y + 33 = 0$$

Podrobnosti o zadané parabole můžeme získat po otevření knihovny Geometry pomocí těchto příkazů:

vertex(p) zobrazí vrchol paraboly,

```
> vertex(p5): coordinates(vertex(p5));
```

$$\left[\frac{-1}{2}, 0\right]$$

focus(p) vrátí ohnisko paraboly,

```
> focus(p5): coordinates(focus(p5));
```

$$[1, 0]$$

directrix(p) vrátí řídící přímku paraboly,

```
> directrix(p5): Equation(directrix(p5));
```

$$x + 2 = 0$$

Equation(p) zobrazí rovnici paraboly.

```
> Equation(p5);
```

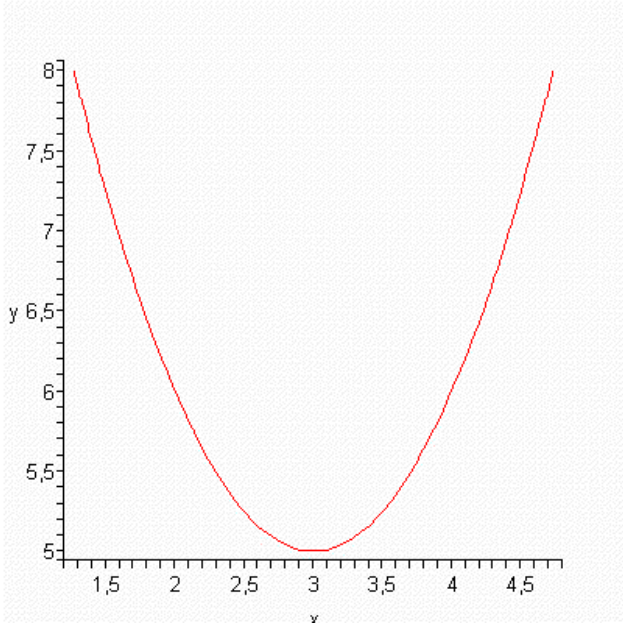
$$-6\,x - 3 + y^2 = 0$$

Příklad:

Zjistěte rovnici paraboly, která bude mít stejné ohnisko jako parabola $y-5=(x-3)^2$ a která bude mít řídící přímku $y=2x+3$.

Možné řešení pomocí programu Maple:

```
> with(plots):
  implicitplot((x-3)^2=y-5, x=1..5, y=4..8);
```



```
> parabola(pp,(x-3)^2=y-5): Equation(pp);
  coordinates(focus(pp));
```

$$x^2 - 6x + 14 - y = 0$$

$$\left[3, \frac{21}{4}\right]$$

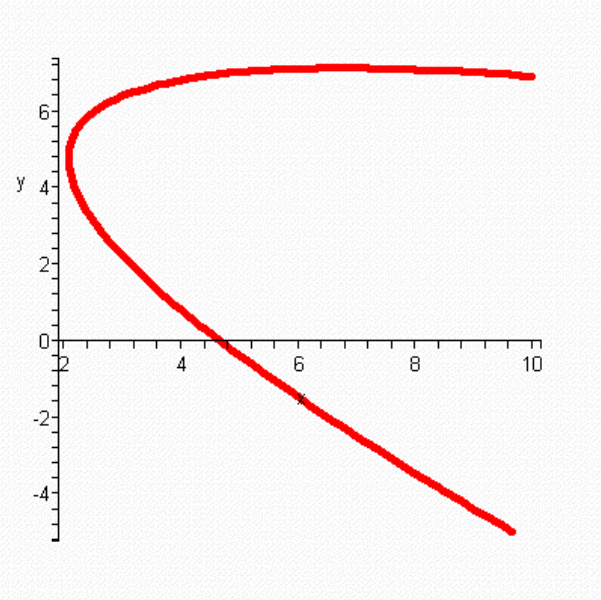
```
> line(l, 2*x+3-y=0, [x,y]);
```

$$l$$

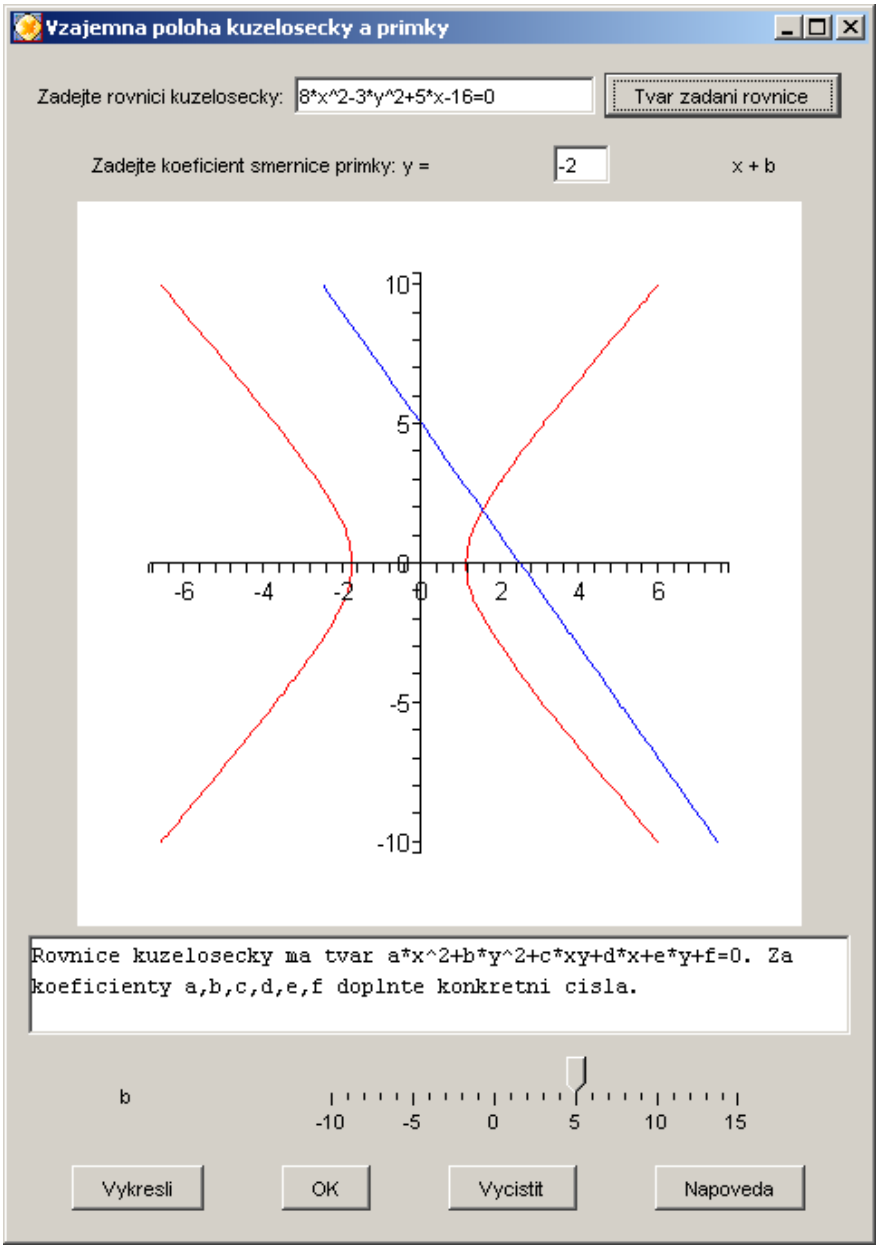
```
> parabola(p, ['directrix'=l, 'focus'=focus(pp)], [x,y]):
  Equation(p);
```

$$\frac{2781}{16} + x^2 + 4xy + 4y^2 - 42x - \frac{93}{2}y = 0$$

```
> with(plots):
  implicitplot(2781/16+x^2+4*x*y+4*y^2-42*x-93/2*y = 0,
  x=0..10, y=-5..10);
```



5.2.7. Vzájemná poloha kuželosečky a přímky



Obr. 5.7 – Vzájemná poloha přímky a kuželosečky (maplet)

Daný maplet po zadání rovnice kuželosečky ve tvaru $ax^2 + by^2 + c \cdot xy + dx + ey + f = 0$. Pro přímku se do rovnice $y = ax + b$ zadává číselný parametr a jako směrnice dané přímky a zvolení parametru b jako posunutí přímky se nastaví na číselném posuvnému. Původní hodnota na posuvnému je nastavena na 0. Po zvolení všech náležitostí a stisknutí tlačítka Vykresli se zobrazí kuželosečka s přímkou. Po jejich zobrazení lze ještě interaktivně měnit posunutí přímky při změně hodnoty na posuvníku b.

Při výpočtu průsečíků přímky s kuželosečkou se vychází z rovnic

$$ax^2 + by^2 + c \cdot xy + dx + ey + f = 0 \qquad (1)$$

$$y = kx + q \qquad (2)$$

Po dosazení (2) do rovnice (1) a úpravě dostáváme kvadratickou rovnici

$$(a + b + ck)x^2 + (d + ek + cq)x + eq + f = 0 \qquad (3)$$

$$A \cdot x^2 + B \cdot x + C = 0 \qquad (4)$$

Počet řešení bude záviset na členu A. Bude-li $A = 0$, pak bude x a y hodnoty

$$x = \frac{-eq - f}{d + ek + cq}$$
$$y = \frac{-fk + dq + cq^2}{d + ek + cq} \qquad (5)$$

Bude-li $A \neq 0$ bude počet řešení bude záviset na diskriminantu kvadratické rovnice (3).

Bude-li $(d + ek + cq)^2 - 4(a + b + ck)(eq + f) < 0$, nebude mít (3) žádný reálný kořen.

Bude-li $(d + ek + cq)^2 - 4(a + b + ck)(eq + f) = 0$, pak bude mít (3) jeden dvojnásobný reálný kořen

$$x = \frac{-d - ek - cq}{a + b + ck}$$
$$y = \frac{-dk - ek^2 + aq + bq}{a + b + ck} \qquad (6)$$

Bude-li $(d + ek + cq)^2 - 4(a + b + ck)(eq + f) > 0$, bude mít (3) dva reálné kořeny a to

$$x = \frac{-d - ek - cq \pm \sqrt{(d - ek - cq)^2 - 4(a + b + ck)(eq + f)}}{2(a + b + ck)}$$
$$y = \frac{-dk - ek^2 + aq + bq \pm \sqrt{(d - ek - cq)^2 - 4(a + b + ck)(eq + f)}}{2(a + b + ck)} \qquad (7)$$

```
> solve({a*x^2+b*x^2+c*x*y+d*x+e*y+f=0, y=k*x+q},{x,y});
```

$$\{x = \text{RootOf}((a+b+k\,c)\_Z^2 + (d+k\,e+q\,c)\_Z + e\,q + f),$$

```
> allvalues(%);
```

$$\left\{ x = -\frac{d+ke+qc-\sqrt{d^2+2dke+2dqc+k^2e^2-2keqc+q^2c^2-4aeq-4af-4beq-4bf-4kcf}}{2(a+b+kc)},\right.$$

$$\left. y = -\frac{k(d+ke+qc-\sqrt{d^2+2dke+2dqc+k^2e^2-2keqc+q^2c^2-4aeq-4af-4beq-4bf-4kcf})}{2(a+b+kc)}+q \right\},$$

$$\left\{ x = -\frac{d+ke+qc+\sqrt{d^2+2dke+2dqc+k^2e^2-2keqc+q^2c^2-4aeq-4af-4beq-4bf-4kcf}}{2(a+b+kc)},\right.$$

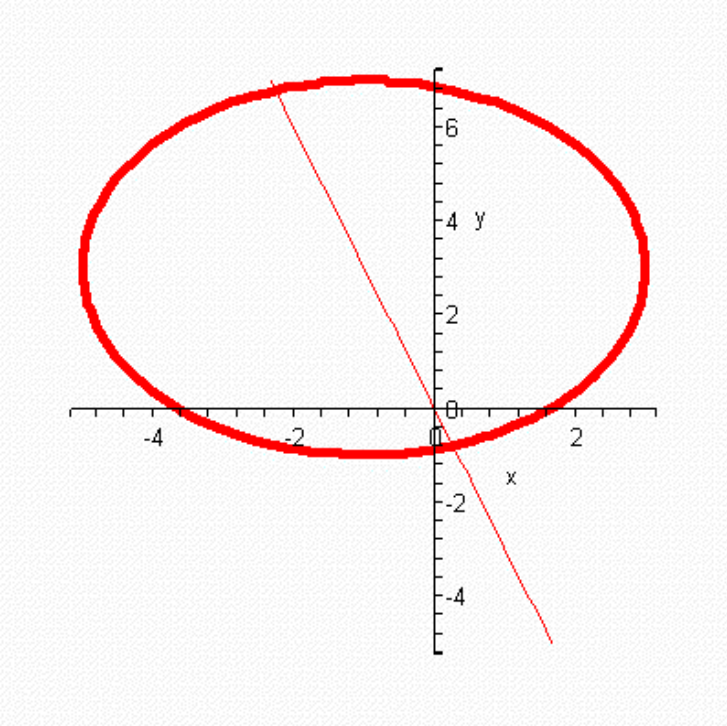
$$\left. y = -\frac{k(d+ke+qc+\sqrt{d^2+2dke+2dqc+k^2e^2-2keqc+q^2c^2-4aeq-4af-4beq-4bf-4kcf})}{2(a+b+kc)}+q \right\}$$

Příklad:

Pro která b, bude mít přímka $p: y = ax + b$ s kuželosečkou $x^2 + 2x + y^2 - 6y - 6 = 0$ dva společné průsečíky, bude-li mít přímka směrnici $a = -3$.

Možné řešení pomocí programu Maple:

```
> with(plots):
  implicitplot({x^2+2*x+y^2-6*y-6=0, y=-3*x},x=-5..7,y=-5..7);
```



```
> solve({x^2+2*x+y^2-6*y-6=0, y=-3*x+b},{x,y});
```

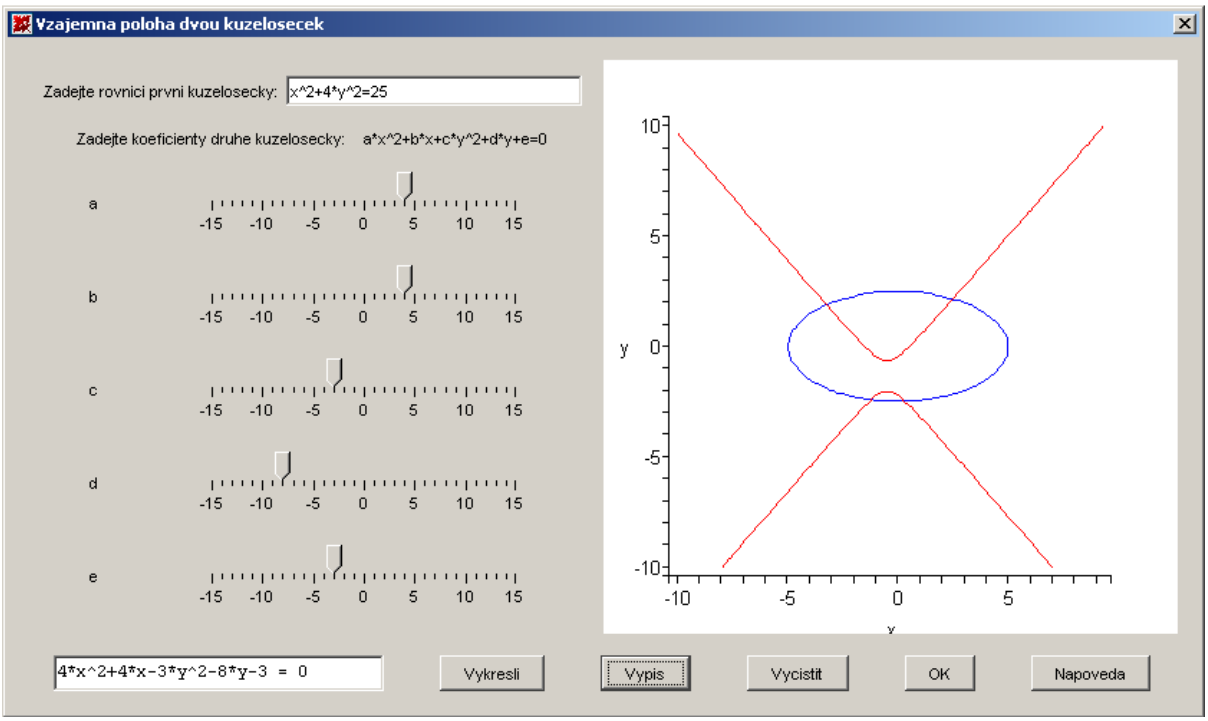
$$\{x = \text{RootOf}(10\_Z^2 + (20 - 6b)\_Z + b^2 - 6b - 6),\ y = -3\,\text{RootOf}(10\_Z^2 + (20 - 6b)\_Z + b^2 - 6b - 6) + b\}$$

```
> solve( {(20-6*b)^2-40*(b^2-6*b-6)>0}, {b} );
```

$$\{-4\sqrt{10} < b,\ b < 4\sqrt{10}\}$$

Chceme-li, aby měla přímka s kuželosečkou dva společné body, musí být diskriminant rovnice $10x^2 + (20-6b)x + b^2 - 6b - 6 = 0$ větší než nula. Diskriminant $(20-6b)^2 - 40(b^2 - 6b - 6) > 0$ upravíme na tvar nerovnice $-4b^2 + 160 > 0$. Z této rovnice dostaneme interval $b \in (-4\sqrt{10}; 4\sqrt{10})$.

5.2.8. Vzájemná poloha dvou kuželoseček



obr. 5.8 – Vzájemná poloha dvou kuželoseček (maplet)

V daném mapletu můžeme jednu rovnici zapsat obecně a v druhé nastavit koeficienty u jednotlivých proměnných v rovnici. Po stisknutí tlačítka ***Vypis*** se vypíše rovnice zadaná pomocí posuvníků. Po stisknutí tlačítka ***Vykresli*** uvidíme vzájemnou polohu obou kuželoseček.

Obecný výpočet vzájemné polohy dvou kuželoseček vychází z jejich rovnic

$$ax^2 + by^2 + c \cdot xy + dx + ey + f = 0$$

$$jx^2 + ky^2 + l \cdot xy + mx + ny + o = 0$$

Obecný výpočet není jednoduchý vzhledem ke stupni rovnice a počtu neznámých. V případě jednodušších rovnic to lze vypočítat. Výpočet jsem uvedla v příkladu.

Příklad:

Jakou vzájemnou polohu budou mít kuželosečky s rovnicemi $x^2+y^2=16, x^2+4y^2=16$? Případně určete průsečíky.

Matematický výpočet:

$$x^2+y^2=16$$
$$x^2+4y^2=16$$

Po odečtení jedné rovnice do druhé dostaneme tuto upravenou rovnici

$$3y^2=0$$
$$y_{1,2}=0$$

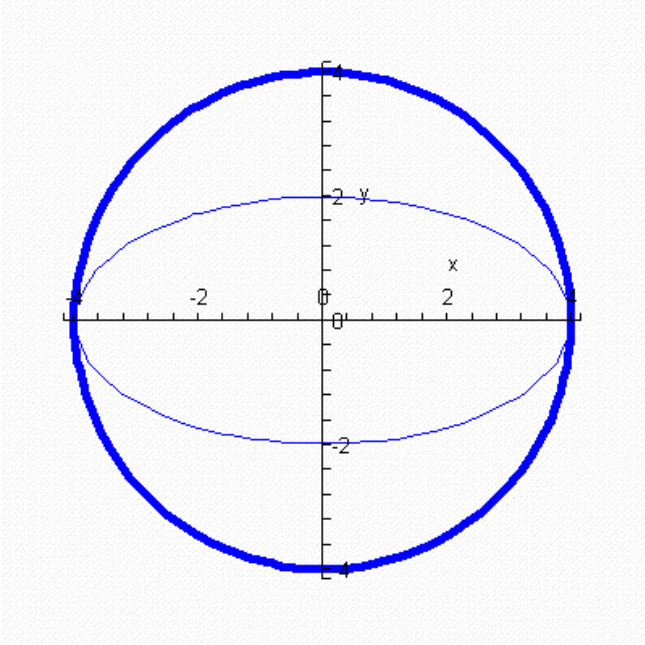
Dosazením tohoto výsledku do původní rovnice dostaneme

$$x^2=16$$
$$|x_{1,2}|=4$$

Dostáváme tedy 4 řešení: $[0,-4],[0,4],[-4,0],[4,0]$

Možné řešení pomocí programu Maple:

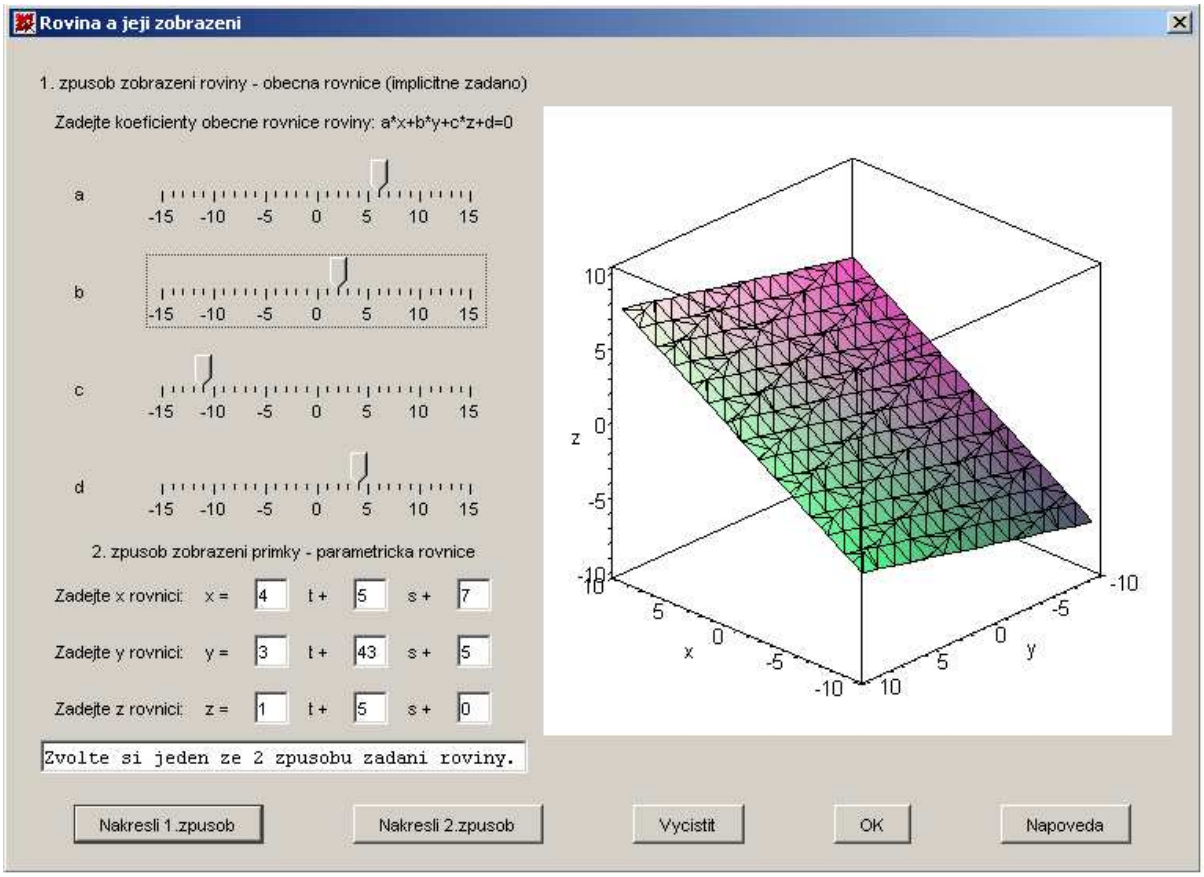
```
> with(plots):
  implicitplot({x^2+y^2=16, x^2+4*y^2=16},x=-5..7,y=-5..7,
  color=(blue,green));
```




```
> solve({x^2+y^2=16, x^2+4*y^2=16},{x,y});
```

{*y* = 0, *x* = -4}, {*y* = 0, *x* = 4}

5.2.9. Rovina a její vyjádření



obr. 5.9 – Rovina a její vyjádření (maplet)

Daný maplet zobrazuje rovinu jak parametricky zadanou, tak obecně zadanou. Po zadání zvoleného způsobu zapsání stiskneme příslušné tlačítko k dané metodě. Poté se nám rovina zobrazí. Stiskneme-li a pohybujeme-li kurzorem v okně grafů, můžeme s rovinou libovolně natáčet.

Maplet ukazuje dva způsoby zapsání rovnice roviny.

- První způsob zapsání rovnice roviny je obecnou rovnicí $ax+by+cz+d=0$. Rovnice je zapsána explicitně. Jak už bylo výše popsáno, do obecné rovnice se píše normálový vektor $\vec{n}=(a,b,c)$.

- Druhý způsob zadání roviny je parametrickou rovnicí. K tomu budeme potřebovat bod (př. A[x,y,z]) a 2 vektory (př. $\vec{u}=(u_1,u_2,u_3)$, $\vec{v}=(v_1,v_2,v_3)$), případně tři body (př. K[k_1,k_2,k_3], L[l_1,l_2,l_3],

M[m_1,m_2,m_3]) nebo dvě přímky zadané parametrickou rovnicí. Parametrická rovnice je ve tvaru $X=A+\vec{u}\cdot t+\vec{v}\cdot s$, popř. $X=K+\overrightarrow{KL}\cdot t+\overrightarrow{LM}\cdot s$, kde KL je vektor z bodu K do bodu L a LM je vektor z bodu L do M. Je-li přímka v prostoru zadána parametricky, bude jejich rovnice vypadat následovně $X=A+\vec{u}\cdot t$ a po rozepsání $x=a_1+u_1\cdot t$, $y=a_2+u_2\cdot t$, $z=a_3+u_3\cdot t$. Obecná rovnice přímky v prostoru neexistuje.

Pokud budeme chtít v prostoru pracovat s objekty, musíme mít otevřenou knihovnu **Geom3d** příkazem *with(geom3d)*. V prostoru budeme pro zobrazení roviny potřebovat definovat tyto objekty: bod, přímka, rovina.

Bod v prostoru je definován příkazem point(název bodu, tři souřadnice bodu). Příklad *point(P, [Px, Py, Pz])* nebo *point(P, Px, Py, Pz)*.

```
> with(geom3d):
  point(A, [0, -1, 2]);point(B, 1, 3, -2);
```

A

B

Podrobnosti o bodu dostaneme pomocí příkazů:

coordinates(P) vrací souřadnice bodu ve tvaru [Px,Py,Pz],

```
> coordinates(A);
```

[0, -1, 2]

xcoord(P) vrátí Px neboli x-ovou souřadnici bodu P,

```
> xcoord(A);
```

0

ycoord(P) vrátí Py neboli y-ovou souřadnici bodu P,

```
> ycoord(A);
```

-1

zcoord(P) vrátí Pz neboli z-ovou souřadnici bodu P.

```
> zcoord(A);
```

2

Přímka v prostoru je definována příkazem line(název přímky, zadání přímky).

line(l, [A, B]) přímka zadaná pomocí dvou bodů

```
> line(l, [A, B]); Equation(l, 't');
```

$$l$$

$$[t, -1 + 4\,t, 2 - 4\,t]$$

line(l, [A, v]) přímka zadaná pomocí bodu a vektoru

```
> point(A,[1,2,3]): v := [7,6,-5]:
  line(l2,[A,v]);
  Equation(l2,'t');
```

$$l2$$

$$[1 + 7\,t, 2 + 6\,t, 3 - 5\,t]$$

line(l, [A, p1]) přímka zadaná pomocí bodu a roviny

```
> point(C,1,2,-1), plane(pp,3*x-5*y+4*z=5,[x,y,z]):
  line(l3,[C,pp]);Equation(l3,'t');
```

$$l3$$

$$[1 + 3\,t, 2 - 5\,t, -1 + 4\,t]$$

line(l, [p1, p2]) přímka zadaná pomocí dvou rovin

```
> plane(p1,4*x+4*y-5*z=12,[x,y,z]):
  plane(p2,8*x+12*y-13*z=32,[x,y,z]):
  line(l5,[p1,p2]);
  Equation(l5,'t');
```

$$l5$$

$$[1 + 8\,t, 2 + 12\,t, 16\,t]$$

*line(l, [a1+b1*t, a2+b2*t, a3+b3*t], t)* přímka zadaná parametrickou rovnicí

```
> line(l6,[1+t, -2-t, t], t): Equation(l5,'t');
```

$$[1 + t, -2 - t, t]$$

Rovina je zadaná příkazem plain(název roviny, zadání roviny).

plane(p, [A, v]) rovina zadaná bodem A a vektorem v

```
> with(geom3d):
  _EnvXName := 'x': _EnvYName := 'y': _EnvZName := 'z':
  point(A, [0, -1, 2]): v := [7,6,-5]:
  plane(p1, [A, v]): Equation(p1);
```

$$16 + 7\,x + 6\,y - 5\,z = 0$$

plane(p, [l1, l2]) rovina zadaná pomocí dvou přímek

```
> line(l1,[1+t, -2-t, t], t): line(l2,[4+3*s,-s,1-s],s):
  plane(p2, [l1, l2]): Equation(p2);
```

$$6 + 2x + 4y + 2z = 0$$

plane(p, [A, B, C]) rovina zadaná třemi body

```
> point(A, [0, -1, 2]): point(B, 1, 3, -2):
  point(C, 5, -2, 0):
  plane(p3, [A,B,C]): Equation(p3);
```

$$24 - 12x - 18y - 21z = 0$$

plane(p, [A, l1, l2]) rovina zadaná bodem a dvěma přímkami

```
> point(K, 5, -2, 0): line(l1,[t, -2+t, t-8], t):
  line(l2,[3*s, -s+4, 1-s], s):
  plane(p4, [A,l1,l2]): Equation(p4);
```

$$12 + 4y - 4z = 0$$

plane(p, eqn, n) rovina zadaná rovnicí roviny

```
> plane(p5, 4*x-3*y-z+3=0, [x,y,z]): Equation(p5);
```

$$4x - 3y - z + 3 = 0$$

Podrobnosti o rovině dostaneme pomocí příkazů:

NormalVector(p) vrátí normálový vektor roviny

```
> plane(p5, 4*x-3*y-z+3=0, [x,y,z]):
  NormalVector(p5);
```

$$[4, -3, -1]$$

Equation(p) vrátí rovnici rovniny

```
> plane(p5, 4*x-3*y-z+3=0, [x,y,z]): Equation(p5);
```

$$4x - 3y - z + 3 = 0$$

Příklad:

Určete rovnici roviny, máme-li zadané dva body A, B se souřadnicemi $A[3,-2,7], B[2,-5,6]$ a víme-li, že rovina má stejný směrový vektor jako přímka tvořená body A, B. Rovina prochází bodem, který je středem úsečky AB.

Možné řešení pomocí programu Maple:

```
> point(A,3,-2,7), point(B,2,-6,5):
  midpoint(C,A,B):
  line(l,[A,B]):
  v := ParallelVector(l);
```

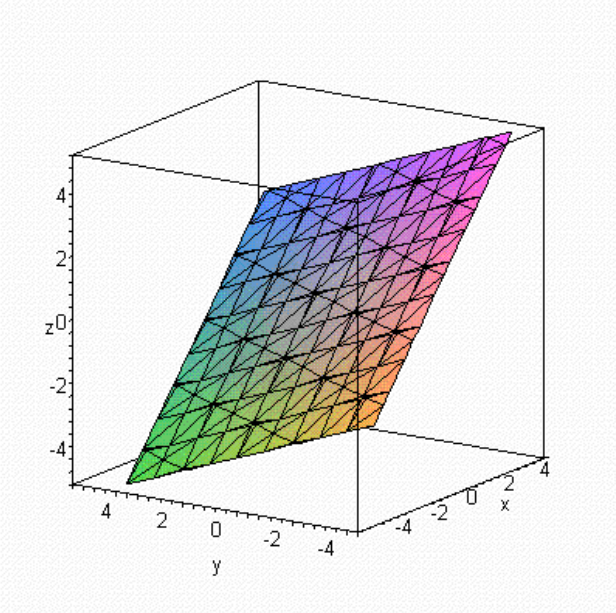
$$v := [-1, -4, -2]$$

```
> plane(p,[C,v]);
  Equation(p,[x,y,z]);
```

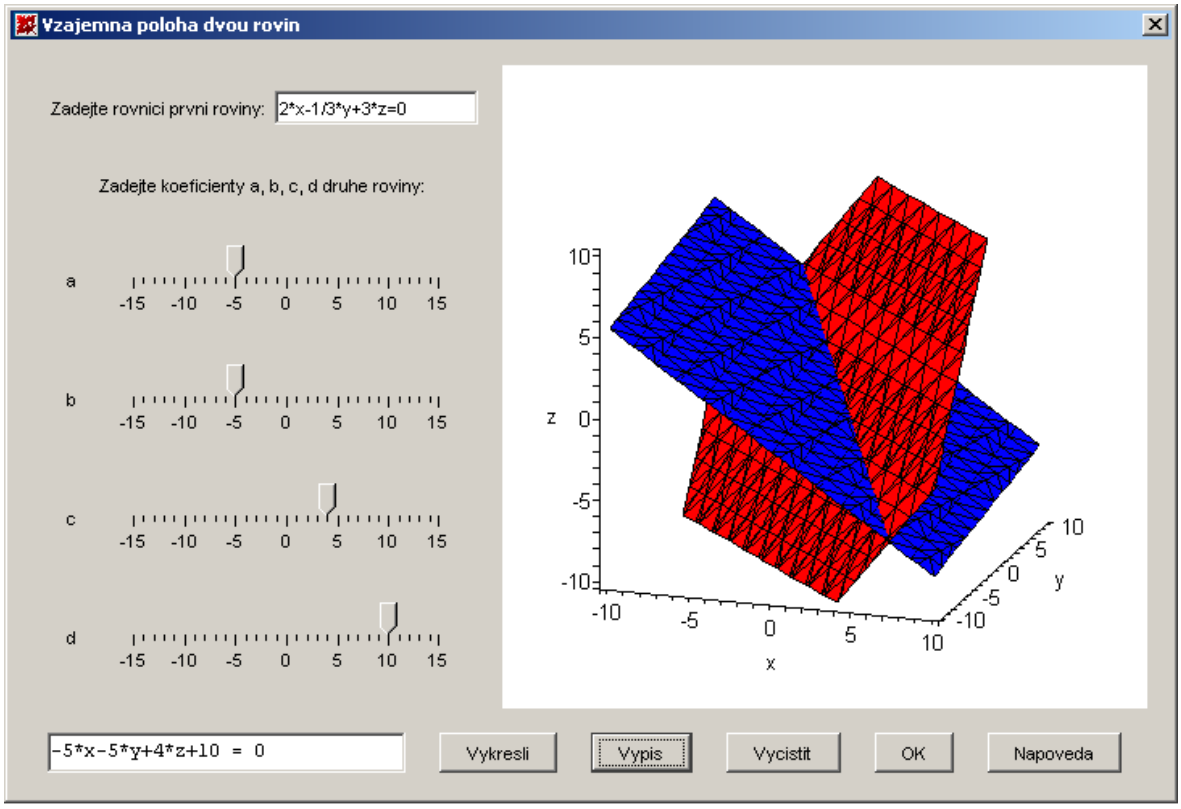
$$p$$

$$-\frac{3}{2} - x - 4\,y - 2\,z = 0$$

```
> with(plots):
  implicitplot3d([-3/2-x-4*y-2*z=0], x=-5..5, y=-5..5,
  z=-5..5, scaling=constrained, axes=boxed);
```



5.2.10. Vzájemná poloha dvou rovin



obr. 5.10 – Vzájemná poloha dvou rovin (maplet)

Daný maplet zobrazuje vzájemnou polohu dvou rovin. První rovinu zapíšeme obecnou rovnicí a u druhé rovnice se můžeme nastavit jednotlivé koeficienty v obecné rovnici roviny. Při změně jednotlivých koeficientů v druhé rovině se bude její poloha automaticky v okně grafů měnit. Tlačítkem ***Vypis*** se nám vypíše obecná rovnice roviny zadané pomocí posuvníků. Tlačítkem ***Vykresli*** se obě roviny zobrazí v okně grafů. Stiskneme-li a pohybujeme-li kurzorem v okně grafů, můžeme s rovinami libovolně natáčet.

Vzájemnou polohu dvou rovin můžeme určit z jejich normálových vektorů z obecných rovnic.

$$\begin{aligned} ax + by + cz + d &= 0 \\ ex + fy + gz + h &= 0 \end{aligned} \qquad (1)$$

Normálový vektor první roviny označíme $\overrightarrow{n_1} = (a,b,c)$, druhé roviny $\overrightarrow{n_2} = (e, f, g)$.

Budou-li normálové vektory lineárně závislé $\overrightarrow{n_2} = k \cdot \overrightarrow{n_1}$, může být vzájemná poloha rovin rovnoběžná nebo totožná. To určíme podle koeficientů d, h.

Bude-li $\overrightarrow{n_2} = k \cdot \overrightarrow{n_1}$ a $d = k \cdot h$, pak budou roviny totožné.

Bude-li $\overrightarrow{n_2} = k \cdot \overrightarrow{n_1}$ a $d \neq k \cdot h$, pak budou roviny rovnoběžné.

Pokud $\overrightarrow{n_2} \neq k \cdot \overrightarrow{n_1}$ budou roviny různoběžné a budou mít společnou průsečnici rovin. Tu vypočítáme z obou rovnic (1) pomocí parametru. Dostaneme parametrické vyjádření přímky v prostoru.

$$x = \frac{(bg - cf) \cdot t + bh - df}{af - eb}$$
$$y = \frac{(ec - ag) \cdot t - ah + ed}{af - eb}$$
$$z = t$$

Příklad:

Určete vzájemnou polohu dvou rovin, víte-li, že první rovina má rovnici $3x - 2y + \frac{1}{3}z - 5 = 0$ a druhá rovina je kolmá na rovinu $-x + 5y + 3z - 7 = 0$ a prochází bodem $M[2,0,-3]$.

Možné řešení pomocí programu Maple:

```
> restart:
  with(geom3d):
  _EnvXName := 'x': _EnvYName := 'y': _EnvZName := 'z':

> plane(p, x+5*y+3*z-7=0, [x,y,z]):
  n:=NormalVector(p);
```

$n := [1, 5, 3]$

```
> point(M, 2,0,-3);
```

$$M$$

```
> plane(q, [M, n], [x,y,z]);
```

$$q$$

```
> Equation(q);
```

$$7 + x + 5\,y + 3\,z = 0$$

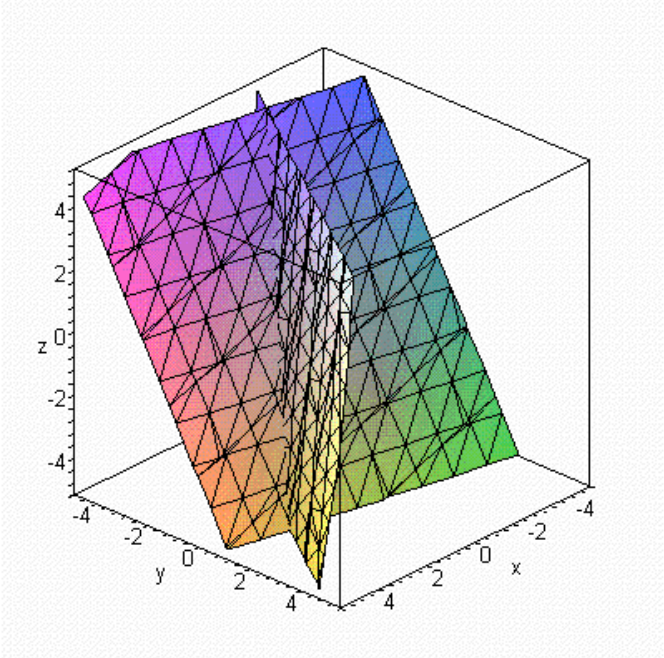
```
> solve({3*x-2*y+1/3*z-5=0,7+x+5*y+3*z = 0},{x,y,z});
```

$$\left\{ z = -3 - \frac{51}{26} y, x = 2 + \frac{23}{26} y, y = y \right\}$$

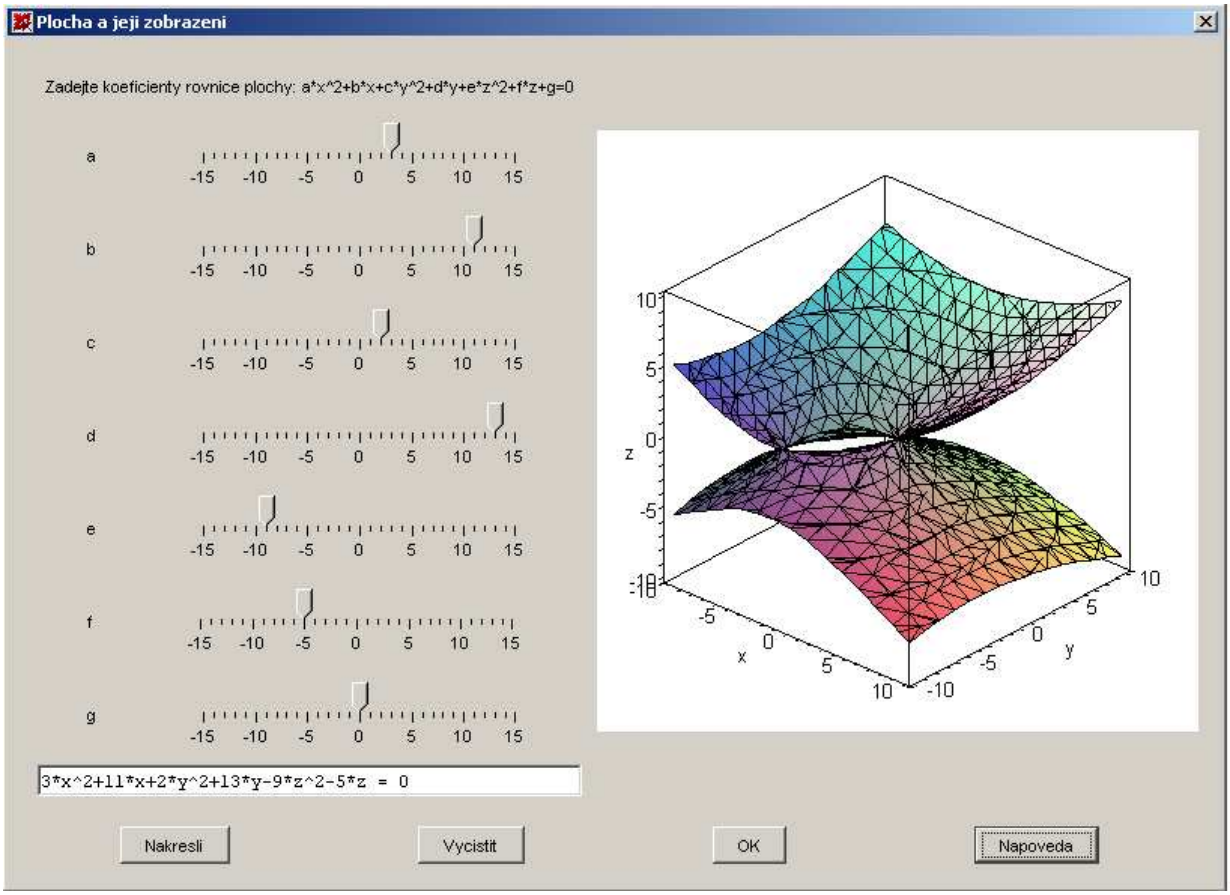
Vidíme, že jsme nalezli společnou průsečnici, roviny jsou tedy různoběžné a průsečnice má rovnici

$$x = 2 + \tfrac{23}{26} t$$
$$y = t$$
$$z = -3 - \tfrac{51}{26} t$$

```
> with(plots):
  implicitplot3d([3*x-2*y+1/3*z-5=0, 7+x+5*y+3*z=0], x=-5..5,
  y=-5..5, z=-5..5, scaling=constrained, axes=boxed);
```



5.2.11. Plocha a její zobrazení



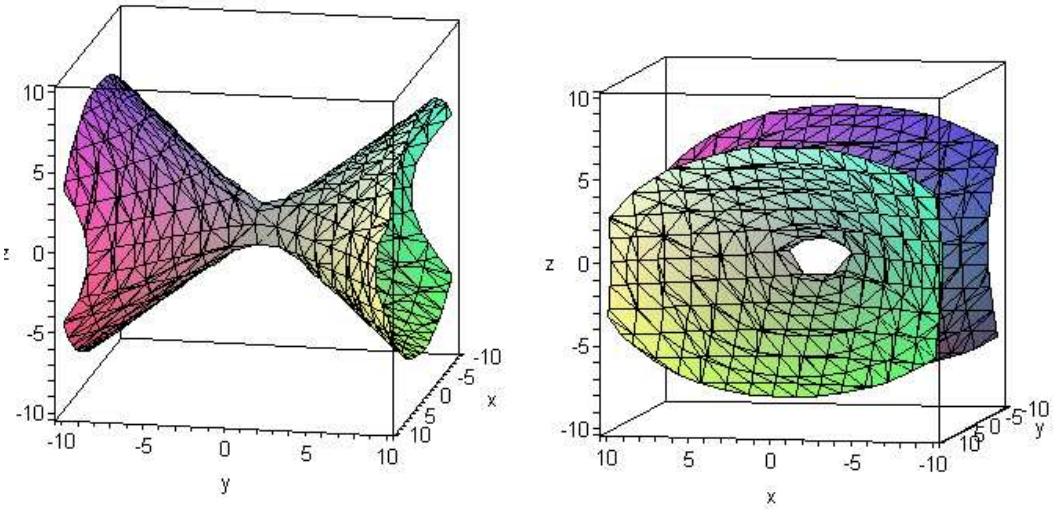
obr. 5.11 – Plocha a její zobrazení (maplet)

V mapletu lze nastavit jednotlivé koeficienty rovnice plochy. Plocha se pak bude zobrazovat v okně grafů. Po stisknutí tlačítka ***Naresli*** se zobrazí zadaná polocha a vypíše se rovnice plochy. Stiskneme-li a pohybujeme-li kurzorem v okně grafů, můžeme s plochou libovolně natáčet.

Rovnice plochy je obecně zadaná rovnicí $ax^2 + by^2 + cz^2 + dxy + exz + fyz + gx + hy + iz + j = 0$. Názvy jednotlivých plochy jsou popsány v kapitole 3.2.

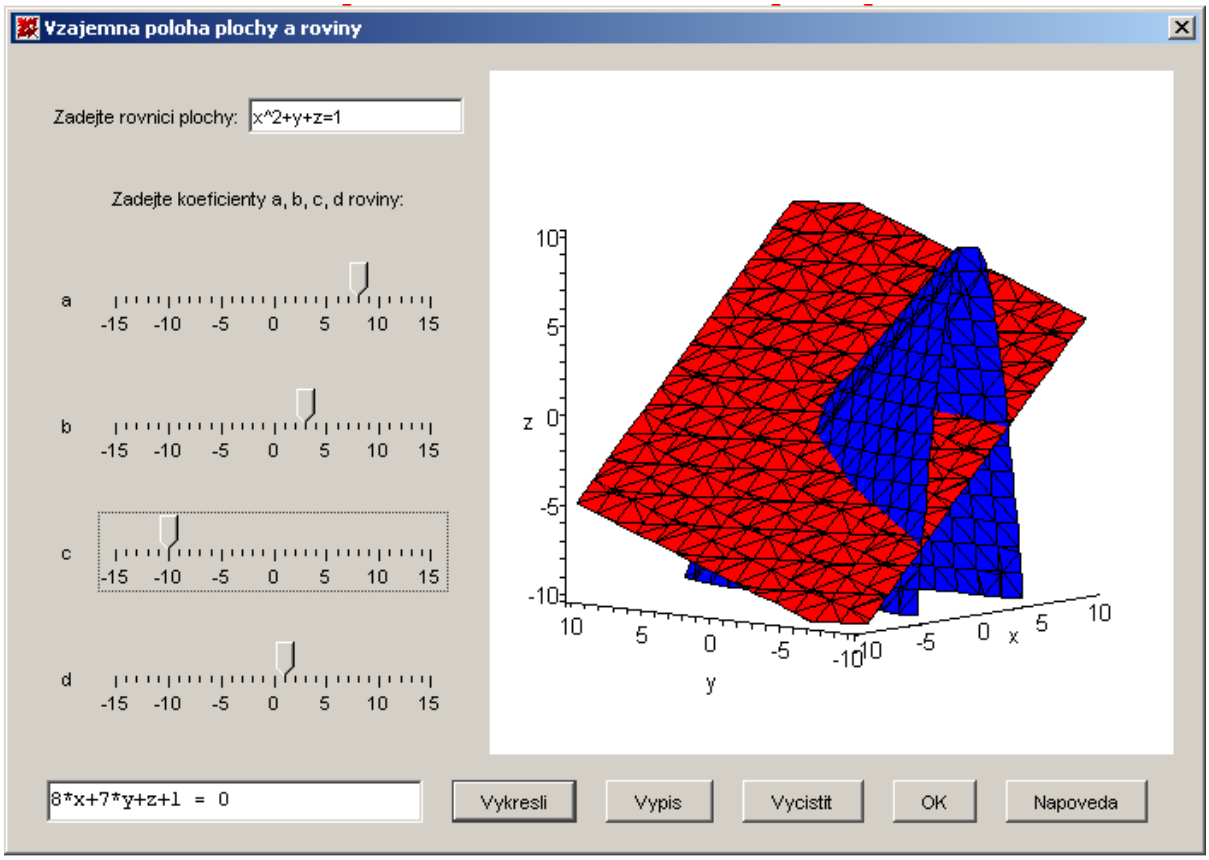
Příklad:

Určete druh plochy, víte-li, že plocha má rovnici $2x^2 - 4y^2 + 4z^2 + 4x - 6y + 3z - 4 = 0$.



Jedná se o jednodílný hyperboloid se středem v bodě $S\left[-1, -\frac{3}{4}, -\frac{3}{8}\right]$ se středovou rovnicí $\frac{(x+1)^2}{\frac{21}{8}} - \frac{(y+\frac{3}{4})^2}{\frac{21}{16}} + \frac{(z+\frac{3}{8})^2}{\frac{21}{16}} = 1$.

5.2.12. Vzájemná poloha plochy a roviny



obr. 5.12 – Vzájemná poloha plochy a roviny (maplet)

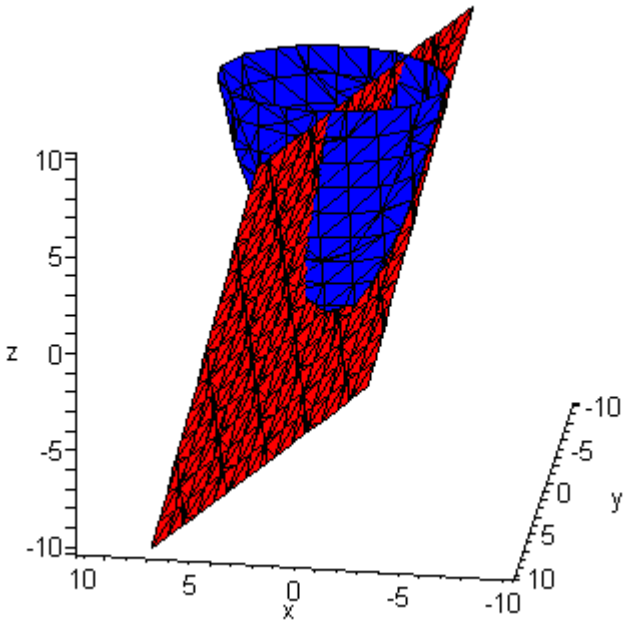
Pro zobrazení vzájemné polohy plochy a roviny musíme zadat plochu pomocí rovnice a pro rovinu zvolit jednotlivé koeficienty. Po stisknutí tlačítka ***Vykresli*** se zobrazí plocha i rovina. Stisknutím tlačítka ***Vypis*** se zapíše rovnice roviny. Stiskneme-li a pohybujeme-li kurzorem v okně grafů, můžeme s plochou libovolně natáčet.

Rovnice plochy je obecně zadaná rovnicí ve tvaru $ax^2 + by^2 + cz^2 + dxy + exz + fyz + gx + hy + iz + j = 0$ Názvy jednotlivých plochy jsou popsány v kapitole 3.2. Rovnice roviny je zadána rovnicí $px + qy + rz + s = 0$.

Pro výpočet můžeme použít Maple, který tyto rovnice vypočítá podle proměnných x, y, z.

Příklad:

Zjistěte vzájemnou polohu plochy a roviny, určenými rovnicemi $2x^2+4y^2-3z^2+x+6y-2z-9=0$ a $-8x+3y-2z+5=0$.



Možné řešení pomocí programu Maple:

```
> solve({2*x^2+4*y^2-3*z^2+x+6*y-2*z-9=0,
  -8*x+3*y-2*z+5=0},{x,y,z});
```

$$\left\{ x = \text{RootOf}(184\_Z^2 + (-144y - 276)\_Z + 78y + 11y^2 + 131),\right.$$

$$\left. z = -4\,\text{RootOf}(184\_Z^2 + (-144y - 276)\_Z + 78y + 11y^2 + 131) + \frac{3}{2}y + \frac{5}{2}, y = y \right\}$$

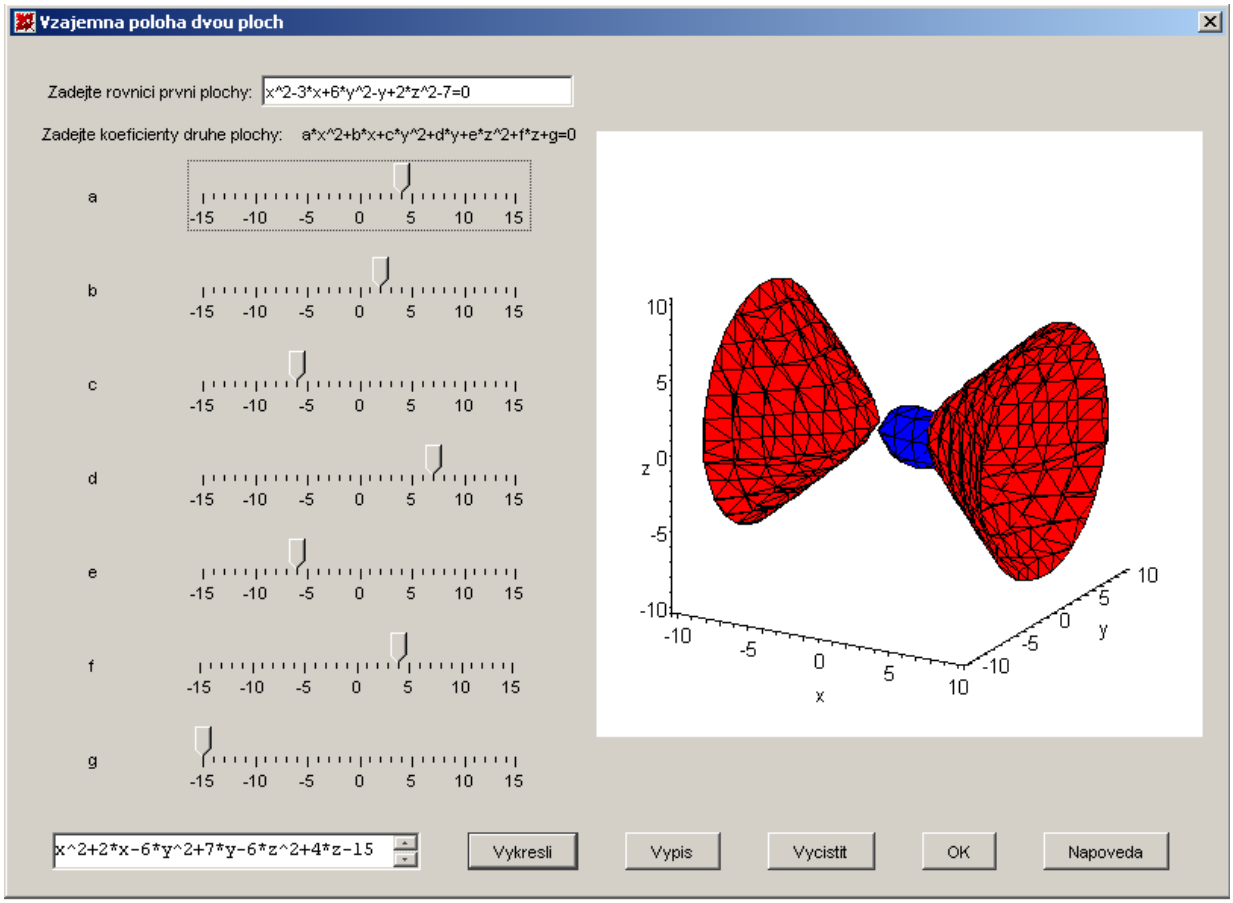
```
> allvalues(%);
```

$$\left\{ y = y, z = -\frac{3}{46}y - \frac{1}{2} - \frac{1}{23}\sqrt{790y^2 + 1380y - 1265}, x = \frac{9}{23}y + \frac{3}{4} + \frac{1}{92}\sqrt{790y^2 + 1380y - 1265} \right\},$$

$$\left\{ y = y, z = -\frac{3}{46}y - \frac{1}{2} + \frac{1}{23}\sqrt{790y^2 + 1380y - 1265}, x = \frac{9}{23}y + \frac{3}{4} - \frac{1}{92}\sqrt{790y^2 + 1380y - 1265} \right\}$$

Vidíme, že řešením je křivka zadaná parametrickými rovnicemi.

5.2.13. Vzájemná poloha dvou ploch



obr. 5.13 – Vzájemná poloha dvou ploch (maplet)

Pro zobrazení vzájemné polohy dvou ploch musíme zadat první plochu pomocí rovnice a pro druhou zvolit jednotlivé koeficienty. Po stisknutí tlačítka ***Vykresli*** se zobrazí vzájemná poloha obou ploch. Stisknutím tlačítka ***Vypis*** se zapíše rovnice plochy zadané pomocí posuvníků. Stiskneme-li a pohybujeme-li kurzorem v okně grafů, můžeme s plochou libovolně natáčet.

Rovnice ploch, u kterých budeme zkoumat jejich vzájemnou polohu jsou

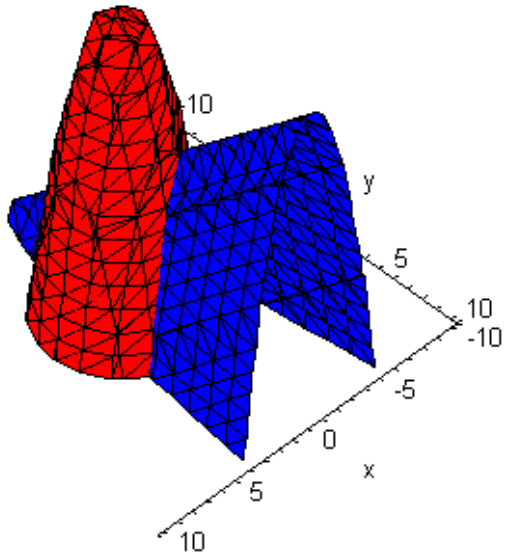
$$ax^2 + by^2 + cz^2 + dxy + exz + fyz + gx + hy + iz + j = 0$$
$$kx^2 + ly^2 + mz^2 + nxy + oxz + pyz + qx + ry + sz + t = 0$$

Příklad:

Zjistěte vzájemnou polohu dvou ploch, které jsou určené těmito rovnicemi $x^2 + y^2 - 2x + 3y + z - 9 = 0$ a $x^2 + x - y + z - 1 = 0$.

Možné řešení pomocí programu Maple:

```
> implicitplot3d({x^2+x-y+z-1=0, x^2+y^2-2*x+3*y+z-9=0},
  x=-10..10, y=-10..10, z=-10..10, numpoints=2000,
  axes=frame);
```




```
> solve({x^2+x-y+z-1=0, x^2+y^2-2*x+3*y+z-9=0},{x,y,z});
```

$$\left\{ x = \frac{1}{3}y^2 + \frac{4}{3}y - \frac{8}{3}, z = -\frac{1}{9}y^4 - \frac{8}{9}y^3 + \frac{61}{9}y - \frac{31}{9} - \frac{1}{3}y^2, y = y \right\}$$

Vidíme, že výsledná křivka je zadána parametricky.

5.3. Využití a začlenění do učiva

Počítače ve výuce

Všeobecné šíření počítačů do všech oblastí lidského života je následováno i využíváním počítačů ve vyučovacím procesu. Z pohledu dnešní didaktiky mají počítače ve vyučování svůj význam vzhledem k tomu, že jsou běžnou součástí života studentů. Zdůrazňuje se především význam interaktivity (aktivní podíl uživatele) a hypermediální povahy (spojení více druhů informací).

Maple ve výuce

Začlenění Maple do výuky matematiky na střední škole není tak výrazně rozšířeno. Myslím si, že je to způsobeno neznalostí učitelů tohoto programu a práce s tímto programem a také vyšší pořizovací cenou.

Moje práce přináší návrh na využití Maple při řešení konkrétních problémů výuky analytické geometrie. Předpokládám použití programu v hodinách spolu s klasickými prostředky výuky jako doplněk výkladu, motivaci na úvod hodiny a pro samostatnou práci studentů. Pro práci s maplety není třeba žádných znalostí s programem Maple. Pro počítání konkrétně zadaných úloh pak již bude zapotřebí seznámit studenty s prostředím Maple, se zadáváním příkazů, představení některých základních příkazů a funkcí. I tyto hodiny jistě přispějí ke zkvalitnění vyučovacího procesu a provázání jednotlivých školních předmětů. Využívání bude jak názorně demonstrační, tak i motivační.

Výuka analytické geometrie v Maple

Program Maple nabízí spoustu funkcí pro ulehčení počítání. Některé důležité příkazy z knihovny **geometry** jsem se snažila popsat u jednotlivých maplet, ve kterých by se dané příkazy mohly hodit pro výpočty příkladů. Používaní těchto příkazů není nikterak složité a jejich názvy jsou logické a snadno zapamatovatelné. Různé příklady jsou uvedeny v nápovědě programu, proto pro využití k počítání stačí znát jen některé z nich. Pomocí programu Maple můžeme ušetřit hodně času, hlavně při řešení složitých rovnic (nerovnic).

Vhodné je i používání grafů k zobrazování objektů jak v rovině, tak i v prostoru.

6. Závěr

Možnosti využití programu Maple 9.5 (případně vyšší verze) ve výuce matematiky jsou velmi široké. Pochopitelně využití má nejen v analytické geometrii, ale i v integrálním a diferenciálním počtu, v algebře, ale i jinde. Myslím, že se mi podařilo zpřístupnit vytváření mapletů dalším uživatelů. Jejich tvorba není nikterak složitá, záleží na konkrétním příkladu.

Stěžejní část mé práce je o tvorbě mapletů, o jednotlivých komponentách, které mohou být v mapletu a o grafickém zobrazení mapletu. Maplety jsou snadno ovladatelné a vizualizují konkrétní vztahy útvarů v analytické geometrii. U popisu každého mapletu je i příklad, k jehož řešení jsem použila příkazy z programu Maple.

Za velmi zdařilou považuji tu část práce, ve které popisuji jednotlivé maplety. Snažila jsem se popsat funkci jednotlivých mapletů, ale i zapsat trochu matematické teorie, vztahující se k danému problému v mapletu. Připsala jsem i příkazy vztahující se k danému apletu. Na konci každého popisu daného mapletu udávám i řešený příklad, ve kterém může použít příkazy uvedené výše.

Nepodařilo se mi tak úplně zachytit všechny chyby v mapletech, které hlásí program Maple, když uživatel zadá neočekávaný vstup pro výpočty či zobrazování. Danému problému bych musela věnovat spoustu času navíc a to jsem nechtěla, neboť si myslím, že je to spíše problém programování než problém matematiky. Doufám, že mi laskavý čtenář a uživatel mapletů odpustí.

Využít jsem toto chtěla pro výuku analytické geometrie na gymnáziu, ale shledala jsem technický problém. Vzhledem k tomu, že program Maple 9.5 je program placený a ne moc rozšířený na středních školách, není nainstalován ani na gymnáziu ve Strakonicích, kde jsem maplety chtěla studenty vyzkoušet. To byl i důvod, kvůli kterému jsem maplety nemohla ani ukázat zdejším

pedagogům. Věřím ale, že škola zakoupí alespoň jednu licenci na program a bude se tak moci nainstalovat alespoň do jednoho notebooku a využít tak spolu s dataprojektorem v hodinách matematiky při probírání analytické geometrie.

Summary

Title: Use of means of the program Maple 9.5 in lessons of analytical geometry

Author: Petra Kubišová

Supervisor: Mgr. Roman Hašek, Ph.D.

In my diploma work I have chosen the topic of application of Maple 9.5 to help students with their study of analytical geometry. They should get knowledge of the meaning of the coeficients in equation of lines, conical sections and planes and solids in space. I gained a lot of useful information at the mathematical conference called Using computers in teaching mathematics which took place in Č. Budějovice in 2005.

The mathematical program Maple allows many calculations. In my diploma work I have made use both of calculating with equations and denoting of graphs. This program also enables its user to program procedures and applications which are called maplets.

The maplets can include parts of the application, which are connected with one another, to show the users relations between the equation and its representation. The maplet can have many parts, for example text fields (input or output), buttons, windows of graphs, buttons of choice or lists or items.

I have prepared maplets for representing of lines, conic sections, planes and solids. In the first part of the prepared maplets the single objects are represented and the second on is about their reciprocal position. Each maplet represents several tasks. Every user can choose different equation of the object of analytical geometry which will be represented. At the end of each maplet there is a particular task with its solution which is possible to compute with the program Maple.

I hope that my diploma work will help everybody who would like to program his or her own maplets and that it will make their effort easier.

Vysvětlivky

Obr. 5.10	obrázek v textu (desátý obrázek v páté kapitole)
Nakresli	název jednotlivých tlačítek v mapletu
solve()	příkazy, které se píší do programu Maple
´ *'thickness'*	parametry pro jednotlivé příkazy
line(název, zobrazení)	obecně zadané příkazy pro představu
Maplets	název knihovny z programu Maple 9.5
`> solve(2*x+3=5);`	příkaz zapsaný v programu Maple 9.5
$\{x = 1, y = -1\}$	výsledek příkazu zobrazený v programu Maple 9.5

Literatura

[1] Heal, K. H.: Maple V, Learning Guide

[2] Hřebíček, a kol.: Solving Probleme in Scietific Computing Using Maple and Matlab

[3] Betounes, D.: Mathematical Computing

[4] Manuály ovládání a programování pogramu Maple 9.5 – v programu

[5] Kočandrle, M., Boček, L.: Matematika pro gymnázia – analytická geometrie, Praha, Prométheus 1996.

[6] Polák, J.: Přehled středoškolské matematiky. Praha, SPN 1991.

[7] Pech, P.: Kuželosečky – skripta, PF JCU, 2003.

[8] Kuřina, F.: Umění vidět v matematice. Praha, SPN 1989.

[9] Osnovy pro čtyřletá gymnázia – matematika.
Praha, MŠ č.j. 20594 / 99 - 22.

[10] Šedivý, J. a kol.: Matematika pro III. ročník gymnázií. Praha, SPN 1986.

[11] Hejný, M. a kol.: Teoria vyučovania matematiky, Bratislava, SPN 1990

[12] Webová stránka společnosti, která vlastní program Maple, je http://www.maplesoft.com

Přílohy na CD

i. Zdrojové kódy vytvořených mapletů

ii. Vytvořené maplety

Seznam jednotlivých mapletů na CD

2.1	Přímka a její vyjádření
2.2	Vzájemná poloha dvou přímek
2.3	Kružnice a její zobrazení
2.4	Elipsa a její zobrazení
2.5	Hyperbola a její zobrazení
2.6	Parabola a její zobrazení
2.7	Vzájemná poloha kuželosečky a přímky
2.8	Vzájemná poloha dvou kuželoseček
2.9	Rovina a její vyjádření
2.10	Vzájemná poloha dvou rovin
2.11	Plocha a její zobrazení
2.12	Vzájemné polohy plochy a roviny
2.13	Vzájemná poloha dvou ploch