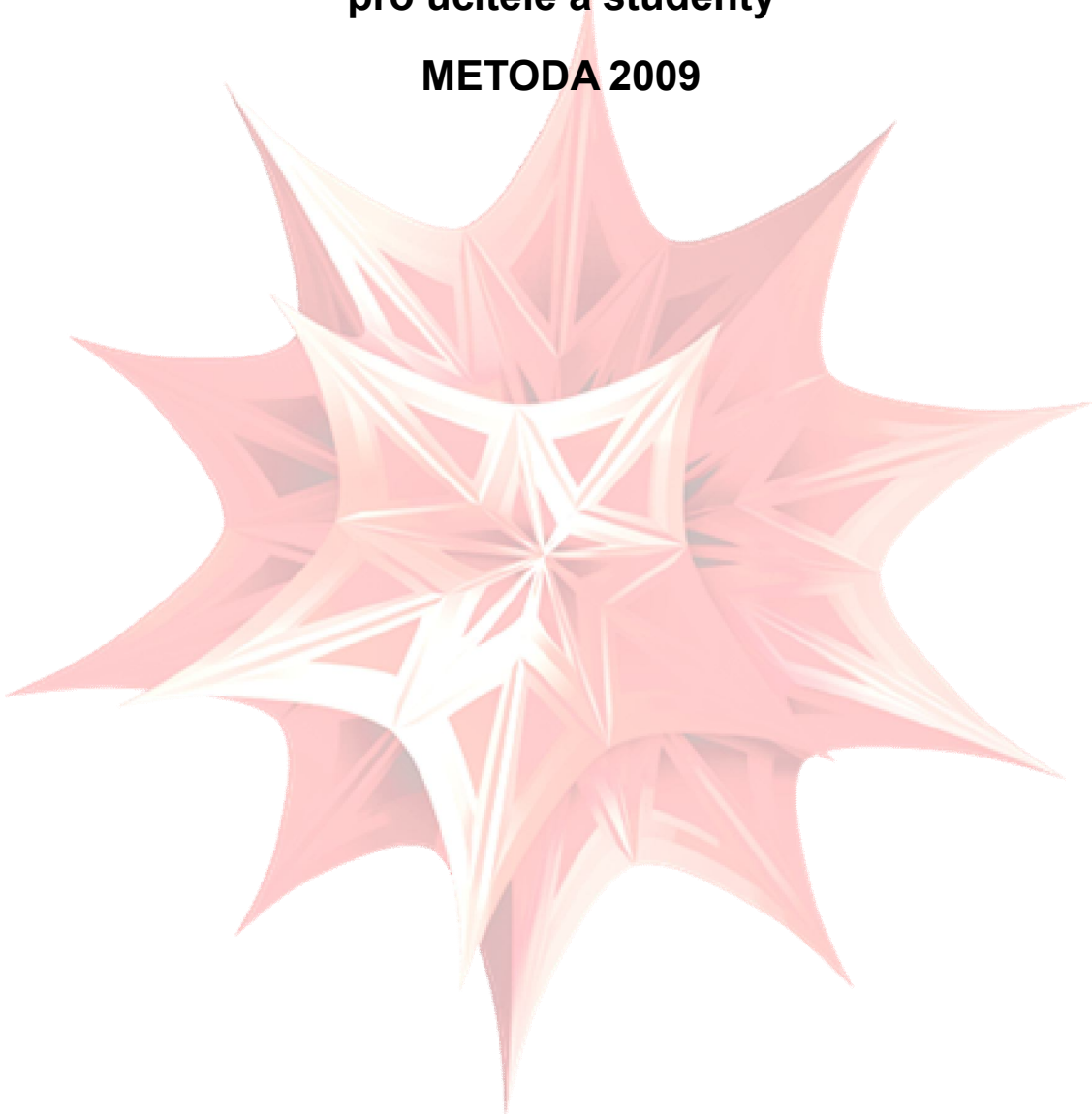
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

MATHEMATICA – příručka s příklady pro učitele a studenty

METODA 2009



Vypracoval: Martin Bouška

Praha 2012

Obsah

1 Úvod

MATHEMATICA je programem pro provádění numerických i symbolických výpočtů a jejich prezentaci dle matematických zvyklostí. Umí pracovat s přibližnými čísly s nastavitelnou přesností, vektory, maticemi, tenzory vyšších řádů, rovnicemi a jejich soustavami, reálnými a komplexními čísly, ale i se symbolickými objekty. Program umožňuje vytvořit písemné, grafické, zvukové a animační výstupy. Pomocí webové služby a formátu CDF lze vytvořit materiály, které mohou být veřejně prezentovány bez nutnosti instalace programu na počítači.

MATHEMATICA je stále živou aplikací. V současnosti využívá i přístupu do internetových databází, obrázků a objektů a neustále rozšiřuje nabídku datových zdrojů. Vzhledem k rozsahu programu není tato aplikace nabízena v různých jazykových mutacích. Všechny příkazy, nápověda a datové zdroje jsou v angličtině.

Program MATHEMATICA začal vytvářet Stephen Wolfram v roce 1986. Pro tento účel založil firmu Wolfram Research a roku 1988 s kolektivem spolupracovníků vytvořil program MATHEMATICA verze 1. V současnosti na vývoji aplikace se podílí asi 1000 programátorů.

Instalační soubor má velikost 920 MB. Po instalaci kompletní verze zabírá 2,84 GB, přičemž 1,4 GB zabírá dokumentace k tomuto systému.

Vlastní aplikaci můžeme rozdělit na tři části:

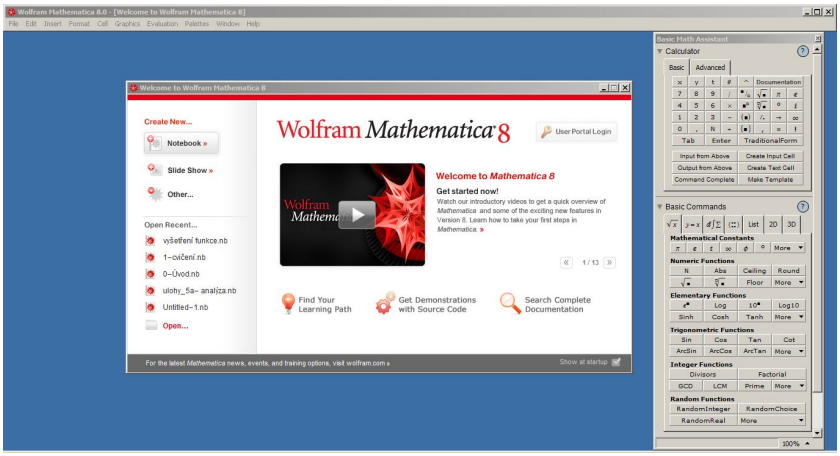
1) **Kernel** – Výpočtové jádro programu, které provádí všechny výpočetní operace. Při chybě během výpočtu nemusíme restartovat celou aplikaci, stačí znovu spustit výpočetní jádro.

2) **Front End** – Vlastní uživatelské prostředí zajišťující komunikaci Kernelu s uživatelem. Základem tohoto prostředí jsou Notebooky, do kterých zapisujeme své příkazy rozdělené na výpočetní buňky. Jedná se o vektorový editor s možností zápisu formátovaného textu. Další variantou jsou SlideShow, notebooky upravené pro prezentaci výpočtů a jejich výsledků.

3) **Packages** – Doplňující systémové knihovny. Protože uživatel běžně využívá 20% schopností aplikace, je vhodné, aby knihovny, které jsou využity jen krátce a výjimečně, zbytečně nezatěžovaly aplikaci a načetly se jen v případě potřeby.

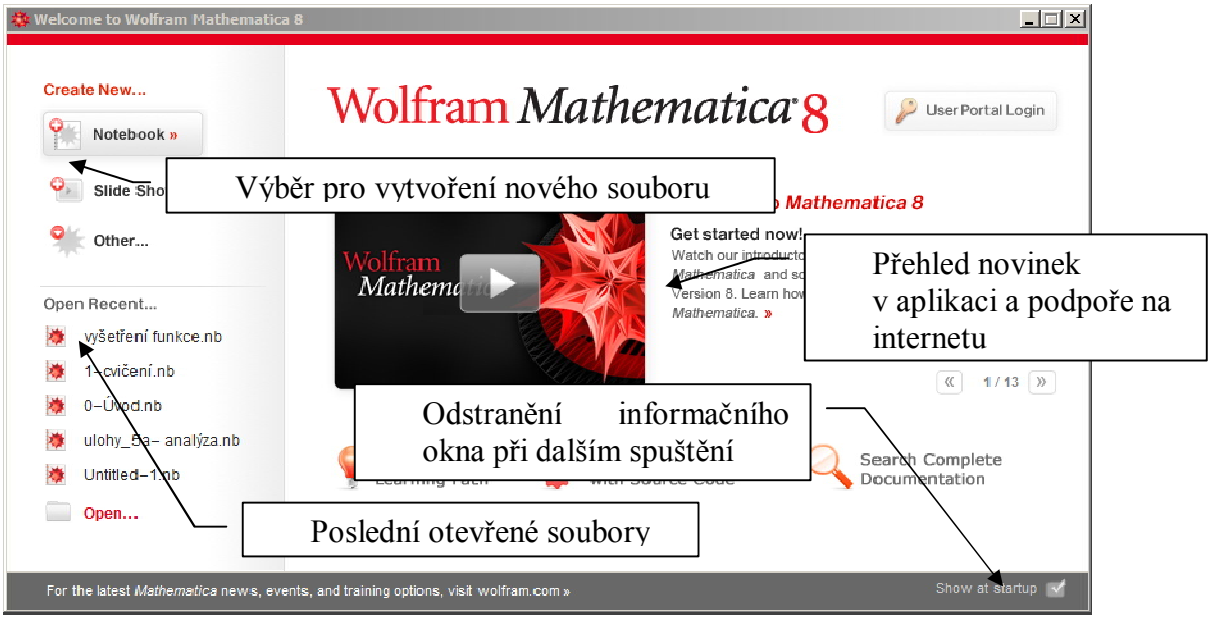
1.1 Aplikace

Při prvním spuštění nám aplikace zobrazí informační okno. Vlastní aplikaci představuje pouze horní menu. V dalších oknech se zobrazují otevřené palety nástrojů.



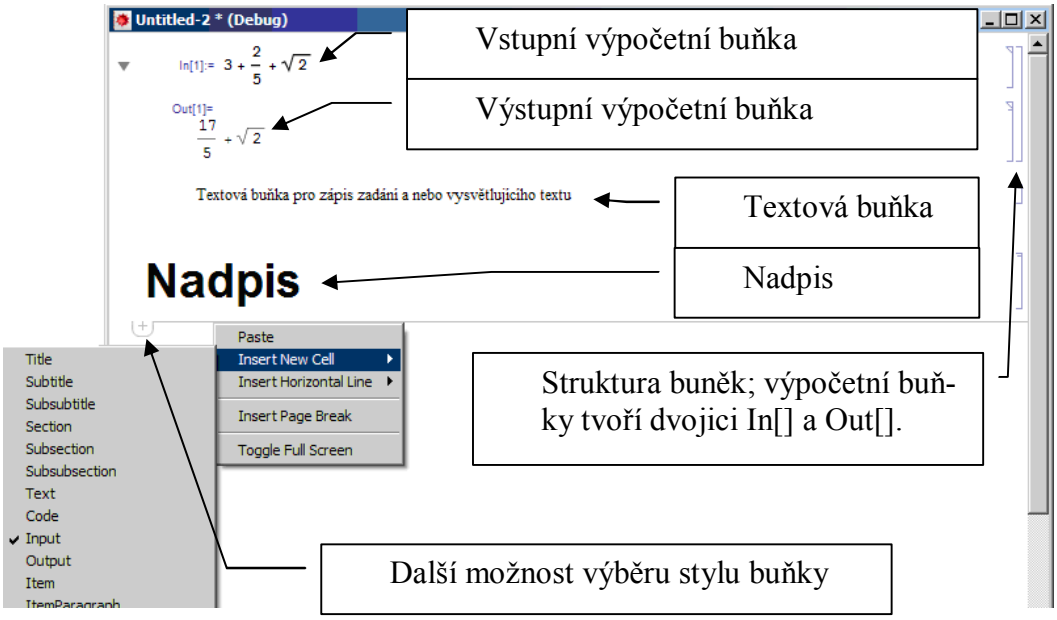
Obrázek 1 – první spuštění aplikace

V uvítacím okně si můžeme prohlédnou odkazy na novinky ve vývoji prostředí aplikace MATHEMATICA a nebo si přímo vybrat z posledních otevřených souborů. V levé horní části vybíráme jaký typ souboru chceme vytvořit. Všechny tyto možnosti jsou dostupné i z hlavního menu nabídky **File**. Pokud si nepřejeme zobrazování tohoto okna při dalším spuštění, odškrtneme v pravém dolním rohu nabídku **Show at startup**.



Obrázek 2 – informační okno

Když otevřeme nový notebook, objeví se prázdný dokument. V okamžiku zápisu začne vytvářet jednotlivé výpočetní a textové buňky, ve kterých je uložen celý obsah našeho dokumentu. Jednotlivé buňky jsou označené na pravé straně hranatými závorkami. Obsahem je určen typ buňky (texty, nadpisy, zadání a výstup). Styl buňky si můžeme zvolit i z lokálního menu. Pokud chceme vytvořit novou buňku, musíme sjet myší na konec dokumentu až na pozici, kdy se kurzor změní na vodorovnou značku. Pravým tlačítkem myši si pak vybereme z lokálního menu styl buňky. Stejné volby nabízí i kliknutí na znaménko + na vodorovné liště za buňkou.



Obrázek 3 – notebook, buňky

Výpočetní buňky tvoří vždy In[*X*] a Out[*X*] dvojici se společným číslem *X*. Zapisovat můžeme pouze do vstupní buňky (In[*X*]). Při každém dalším přepočítání se číslo *X* mění a určuje počet výpočtů od spuštění Kernelu. Spuštění výpočtu Kernelu provedeme umístěním kurzoru do výpočetní buňky a stisknutím klávesnice Shift+Enter, nebo Enter na numerické klávesnici. Pokud jsme výpočetní výraz omylem zapsali do textové buňky, výpočet se neprovede. Změnu stylu však můžeme provést z lokálního menu příkazem **Convert To**, nebo z menu po kliknutí na závorku na pravé straně dokumentu.

Příklad 1

Vyzkoušejte si první výpočet v aplikaci. Vyberte styl vstupní buňky Input (např. v lokálním menu) a zadejte: 5+3 a zapněte výpočet (Shift+Enter).

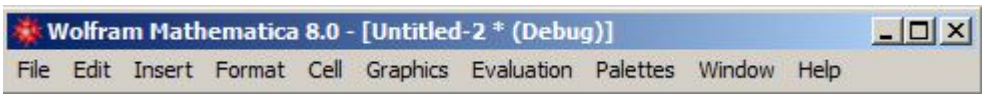
Obrázek 4 – Příklad 1 – výpočet

Pokud výraz zadáme chybně, aplikace oznámí chybu a my můžeme zadání opravit. Zadejme chybný výraz 5+ a proveďme výpočet. Rámeček buňky se při chybném zadání označí červeně. Pokud nevidíme chybové hlášení, stačí kliknout na žlutý symbol + v pravém okraji.

Obrázek 5 – Příklad 1 – chyba

1.2 Menu

Vlastní aplikace je pouze hlavní nabídka. Ostatní okna můžeme samostatně zavírat a otvírat. Při zavření menu se aplikace ukončí. Všechny nabídky tohoto programu jsou anglicky, proto je vhodná aspoň pasivní znalost anglického jazyka. Jazykové mutace menu existují, ale podléhají licenčním podmínkám.



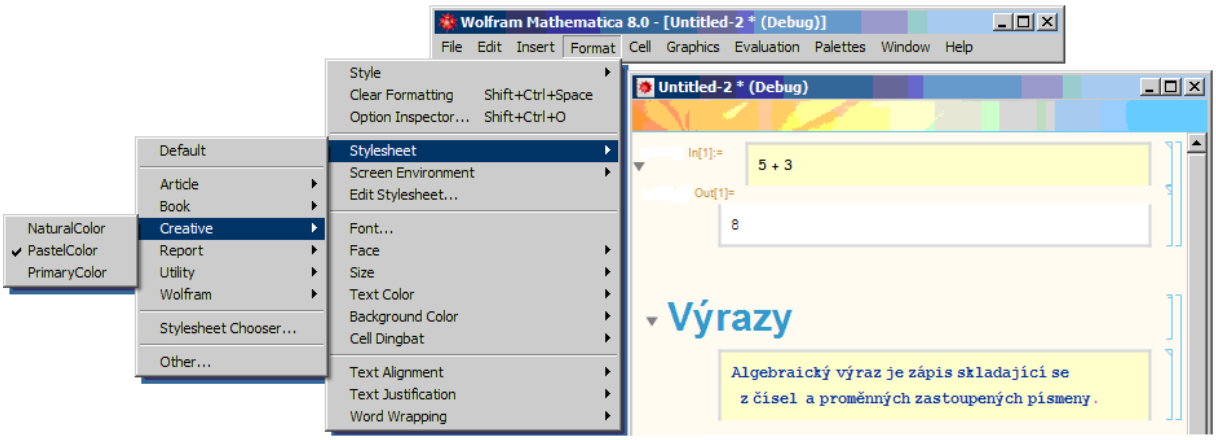
Obrázek 6 – hlavní menu

V nabídce **File** nalezneme všechny možnosti související s tvorbou nových dokumentů, ukládáním dokumentů, zavíráním oken, tištěním dokumentů, přehledem posledních souborů a ukončením aplikace.

Edit obsahuje položky pro práci se schránkou a vyhledáváním. Poslední položkou této nabídky jsou **Preferences**. Tou nastavíme vlastnosti menu a vlastnosti zobrazení výrazů v notebooku.

Insert je nabídka určená pro jednoduché vkládání objektů a složitějších matematických konstrukcí (mocnin, odmocnin, matic, tabulek,...). Zde najdeme i klávesové zkratky těchto voleb. Ty využijeme především při textovém zápisu výrazů.

Položka **Format** je zaměřená na nastavení formátu buněk, formátu použitého písma, zarovnání textu a stylu sešitu v podnabídce **StyleSheet**.



Obrázek 7 – nastavení stylu notebooku

Cell je nabídka činností pro zacházení s buňkami. Ty můžeme seskupovat, slučovat, rozdělovat, pojmenovávat, nebo odstranit.

Graphics je výběrem činností s grafickými objekty. Především jejich umístění a zarovnání.

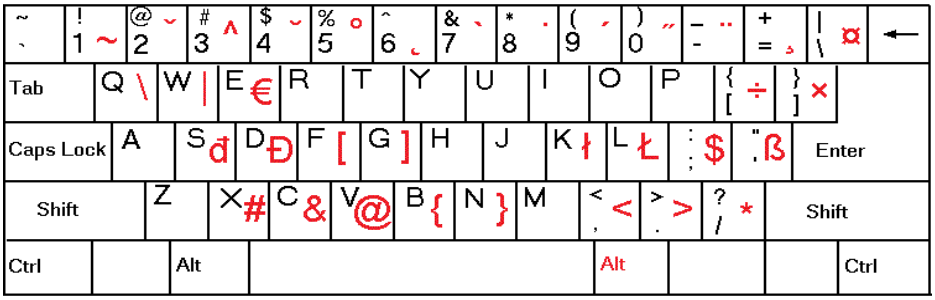
Položka označená **Evaluation** slouží k nastavení výpočtů buněk. Zde lze zapnout a vypnout výpočetní jádro aplikace Kernel, krokovat výpočty a přepočítat vybrané buňky. Při prvním spuštění je zaškrtnutá volba Debugger (ladění). Tato volba je označena v příkladech slovíčkem Debug v popisku buněk.

Nabídka **Palettes** slouží k obsluze tématických palet. Ty nám pomohou lépe zapisovat matematické formule a objekty, pokud nepreferujeme řádkový zápis matematických formulí.

Standardním typem nabídky je položka **Window** a **Help**, ty obsahují funkce uspořádání oken aplikace a podpůrné prostředky, včetně dokumentace, virtuální knihy, helpu a internetových podpůrných zdrojů. **Internet Conectivity** je položka z menu **Help**, která je důležitá pro nastavení připojení k internetu. Program MATHEMATICA spoustu údajů dohledává v externích databázích na internetu. Pokud je tato položka špatně nastavená, některé nástroje programu přestanou správně pracovat.

1.3 Základní práce a důležité dohody

Program MATHEMATICA je case-sensitive. To znamená, že rozlišuje malá a velká písmena. Všechny funkce a příkazy začínají velkým písmenem, ostatní písmena jsou malá (např. Sin[], Plot[], ...). Pokud je název funkce složený z více slov, je vždy první písmeno jednotlivých slov velké (např. ChemicalData[], ListPlot[], ...). Parametry těchto příkazů se vkládají do hranatých závorek. Složené závorky se používají pro seznamy a tabulky. Kulaté závorky nahrazují závorky v matematických výrazech. Pokud užíváme českou klávesnici, mohou být pro nás některé znaky hůř dostupné. V takovém případě využijme málo známé kombinace klávesových zkratek s pravým tlačítkem Alt.



Obrázek 8 – možnosti speciálních znaků na české klávesnici

Nejjednodušší způsob použití programu je vědecká kalkulačka. Nezapomeňme, že zde platí anglická konvence a čísla se zapisují s desetinnou tečkou! Pro základní funkce kromě předdefinovaných formátů z palety **Basic Math Assistant** můžeme použít i řádkové příkazy a symboly, které známe i z vědeckých kalkulaček.

Základní matematické operace zapisujeme:

a^*b*	umocňování,	nejvyšší priorita;
a / *b*	dělení,	střední priorita;
a * *b*	násobení,	střední priorita;
a + *b*	sčítání,	nejnižší priorita;
a – *b*	odčítání,	nejnižší priorita.

Prioritu měníme použitím závorek, jak jsme v matematice zvyklí. V ostatních případech výpočtu algebraických výrazů platí pravidlo vyhodnocování z pravé strany.

Příklad 2

Vypočítejte hodnoty výrazů a porovnejte jejich výsledky. Zdůvodněte, proč se výsledky od sebe liší.

3*2^3+1*2;

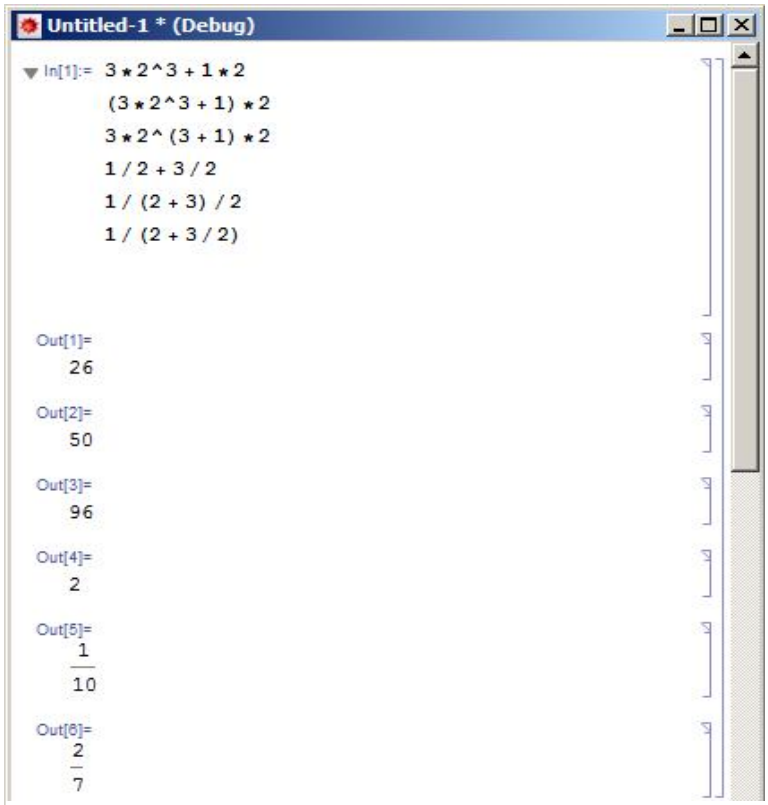
(3*2^3+1)*2;

3*2^(3+1)*2;

1/2+3/2;

1/(2+3)/2;

1/(2+3/2).



Obrázek 9 – Příklad 2

Výsledky jsou po řadě 26, 50, 96, 2, $\frac{1}{10}$, $\frac{2}{7}$. Hodnoty jsou rozdílné z důvodu priority operací. V prvním příkladě má nejvyšší prioritu mocnina, pak jednotlivá násobení a nakonec sčítání. V druhém a třetím výrazu mění prioritu zpracování závorky. Podobný systém priorit platí i u čtvrtého, pátého a šestého výpočtu.

Základní relační operátory jsou:

<	menší,
>	větší,
<=	menší nebo rovno,
>=	větší nebo rovno,
==	je rovný (stejné hodnoty),
===	je stejný (stejná hodnota i typ proměnné),
!=	je různý.

```
In[1]:= 2 / 5 < 3 / 8
Out[1]=
  False

In[2]:= 2 + 5 == 7
Out[2]=
  True
```

Obrázek 10 – relační operátory

Výsledkem těchto operátorů je odpověď True (pravda), nebo False (nepravda) podle platnosti operátoru. Pozor program MATHEMATICA počítá s přesnými čísly. Podle nastavení zobrazuje výsledky s daným počtem desetinných míst, ale Kernel pracuje s přesnou hodnotou. Proto stejně zobrazená čísla nemusí mít stejnou hodnotu.

```
In[1]:= 2 / 7 - 0.6
Out[1]=
  -0.314286

In[2]:= (2 / 7 - 0.6) == -0.314286
Out[2]=
  False
```

Obrázek 11 – přesnost čísel

Přesnost zobrazení čísel eventuálně nastavíme v nabídce **Edit > Preferences** na záložce označené **Appearance / Numbers Formatting** u vlastnosti **Displayed precision**. Pokud potřebujeme zobrazit výsledek se stanoveným počtem desetinných míst, použijeme funkce N[]. Tu můžeme použít klasicky se zápisem parametrů do hranatých závorek, nebo jako příkaz takzvané roury, kdy název funkce zapíšeme za výraz a dvě lomítka.

```
In[4]:= N[2 / 3 - 1 / 2]
Out[4]=
  0.166667

In[5]:= 2 / 3 - 1 / 2 // N
Out[5]=
  0.166667
```

Obrázek 12 – numerický přepočet výrazu ve tvaru funkce a roury

Jako každá dobrá vědecká kalkulačka má i tento program celou řadu matematických funkcí, v tomto případě jich je asi 1700. První skupinou těchto funkcí jsou trigonometrické a cyklometrické funkce:

Sin[], Cos[], Tan[], Cot[] sin x, cos x, tg x, cotg x.

ArcSin[], ArcCos[], ArcTan[], ArcCot[].

Další význačné funkce:

Abs[]	absolutní hodnota,
Exp[x]	e^x,
Log[x]	ln x,
Log[b, x]	$\log_b x$,
Power[b,x]	b^x,
Sqrt[]	druhá odmocnina.

Kromě toho obsahuje MATHEMATCA řadu konstant:

Degree	konstanta na převod stupňů na radiány (π/180),
E	e,
I	komplexní jednotka,
Infinity	∞,
Pi	π, lze ho zapsat i znakem π (pořadím kláves Esc, p,Esc).

Příklad 3

Vypočítejte hodnotu sin 30°, cos(π/4) a 1/∞. Zobrazte číslo π a e s přesností na 50 míst.

```
In[1]:= Sin[30 Degree]
Out[1]= 1/2

In[2]:= Cos[Pi / 4]
Out[2]= 1/Sqrt[2]

In[3]:= 1 / Infinity
Out[3]= 0

In[4]:= N[Pi, 50]
Out[4]= 3.1415926535897932384626433832795028841971693993751

In[5]:= N[E, 50]
Out[5]= 2.7182818284590452353602874713526624977572470937000
```

Obrázek 13 – Příklad 3

Na vědeckých kalkulačkách občas používáme předchozí výpočet. Protože Kernel si také indexuje výsledky, můžeme se na předchozí výsledek odkázat znakem %, nebo na vybraný výpočet pomocí indexu symbolem % *X*, kde *X* představuje číslo výpočetní buňky.

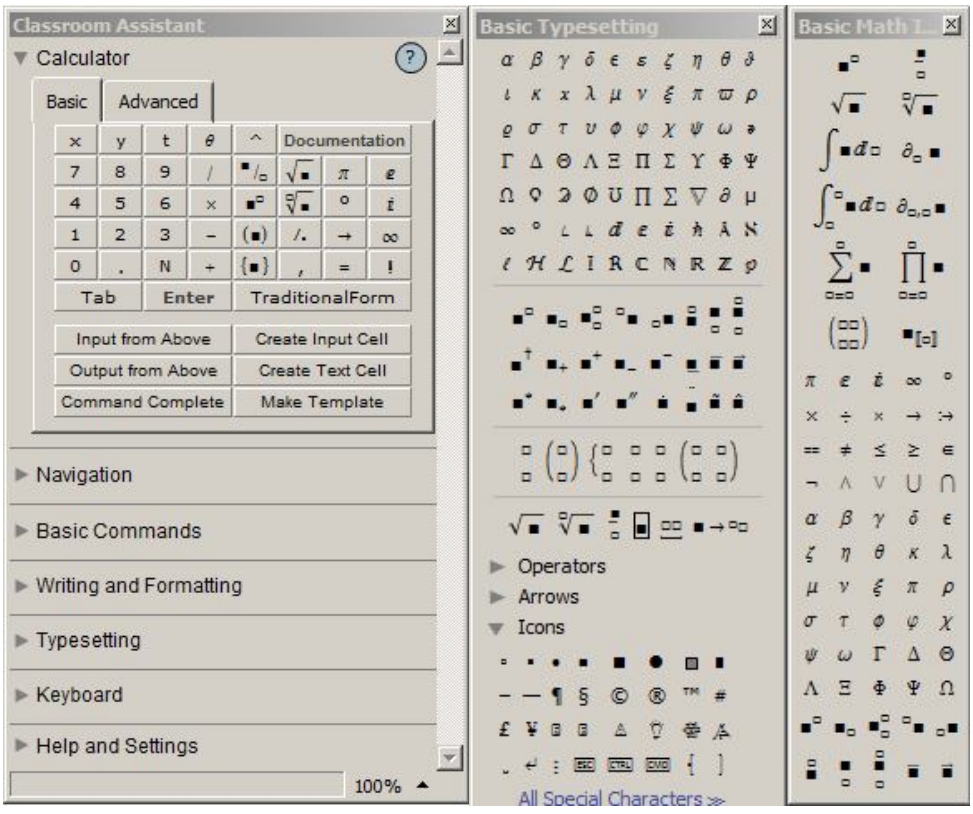
Příklad 4

Vypočítejte odmocninu z 18 a v následujícím výpočtu přičtěte odmocninu ze 2. V další buňce vynásobte hodnotu výsledku z první buňky odmocninou z 8. (Poznámka: Před prvním výpočtem resetujte Kernel, aby indexace buněk nepokračovala, ale probíhala znovu od jedné, použijte příkazů z nabídky **Evaluation > Quit Kernel > Local**). Na tamto obrázku je zobrazena na pravé straně struktura výpočetních buněk. Pro další příklady nemá tato struktura hlubší význam, proto ji dále nebudeme zobrazovat.

```
In[1]:= Sqrt[18]
Out[1]=
        3 √2

In[2]:= % + Sqrt[2]
Out[2]=
        4 √2

In[3]:= %1 * Sqrt[8]
Out[3]=
        12
```

Obrázek 14 – Příklad 4

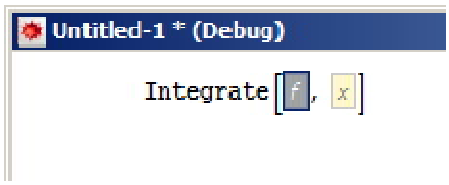
Všechny tyto operace zapisujeme pomocí palet z nabídky **Palettes** v hlavním menu. Na obrázku vidíme paletu Classroom Assistant a palety z nabídky Other, které můžeme znát ze starších verzí programu.



Obrázek 15 – paleta Classroom Assistant a palety z nabídky Other

1.4 Nápověda

Program MATHEMATICA má poměrně rozsáhlý systém aktivní nápovědy. Celá nápověda je vytvořen pomocí aktivních notebooků. Ukázkové příklady tedy lehce zkopírujeme do své aplikace a upravíme podle svých požadavků. Nápovědu k příkazu vyvoláme označením funkce v textu a stisknutím klávesy F1. Pokud neznáme celý název funkce, stačí napsat několik úvodních písmen z názvu funkce a stisknout kombinaci kláves Ctrl+K. Objeví se rozbalovací seznam příkazů se shodným začátkem názvu. Po výběru názvu můžeme zobrazit parametry funkce kombinací kláves Shift+Ctrl+K.

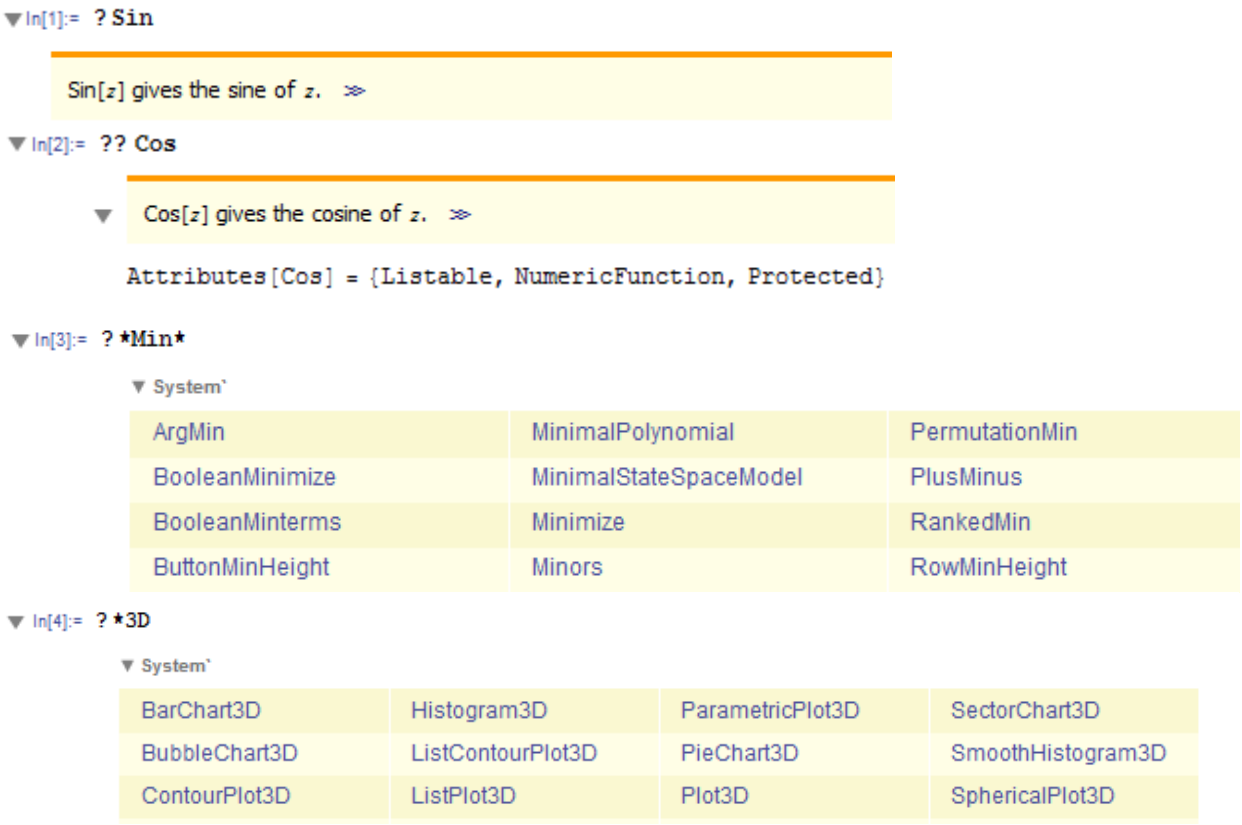


Obrázek 16 – nápověda názvu Ctrl+K **nápověda parametrů Shift+Ctrl+K**

Jednoduchý dotaz na nápovědu k funkci můžeme provést i pomocí kombinace znaků ? a *.

Příklad 5

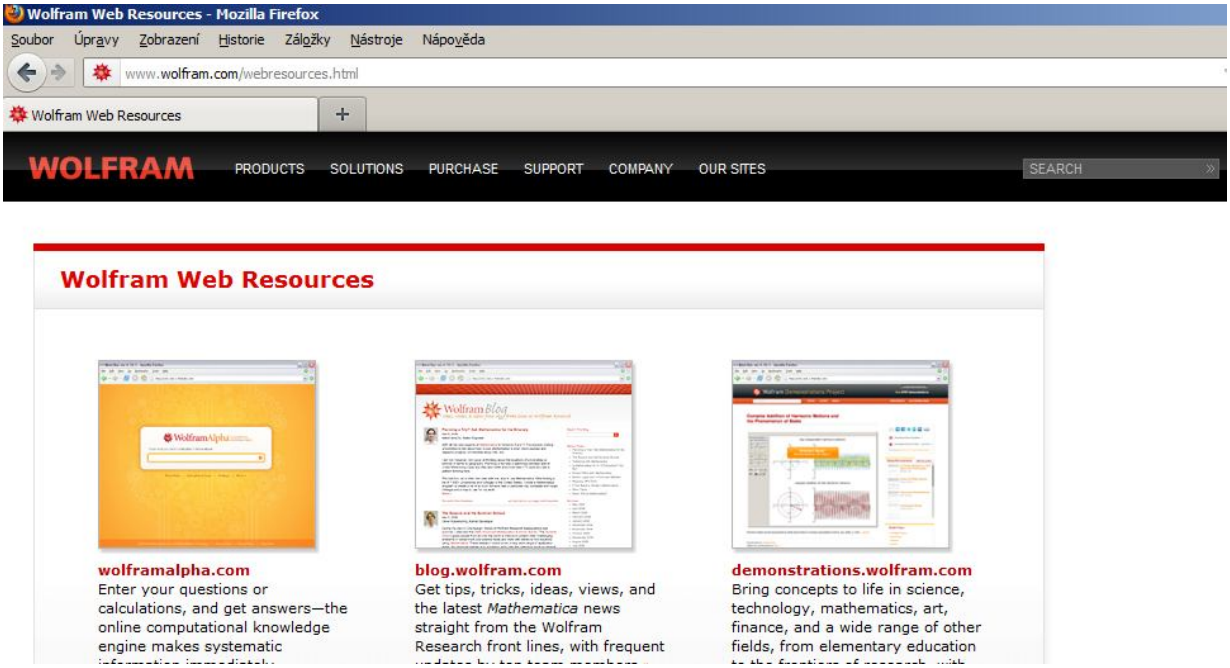
Zobrazte základní nápovědu k funkci Cos[], podrobnou nápovědu k funkci Sin[], vyhledejte funkci, která v názvu obsahuje písmena Min a funkce končící na písmena 3D. Výsledek je na následujícím obrázku.



Obrázek 17 – Příklad 5

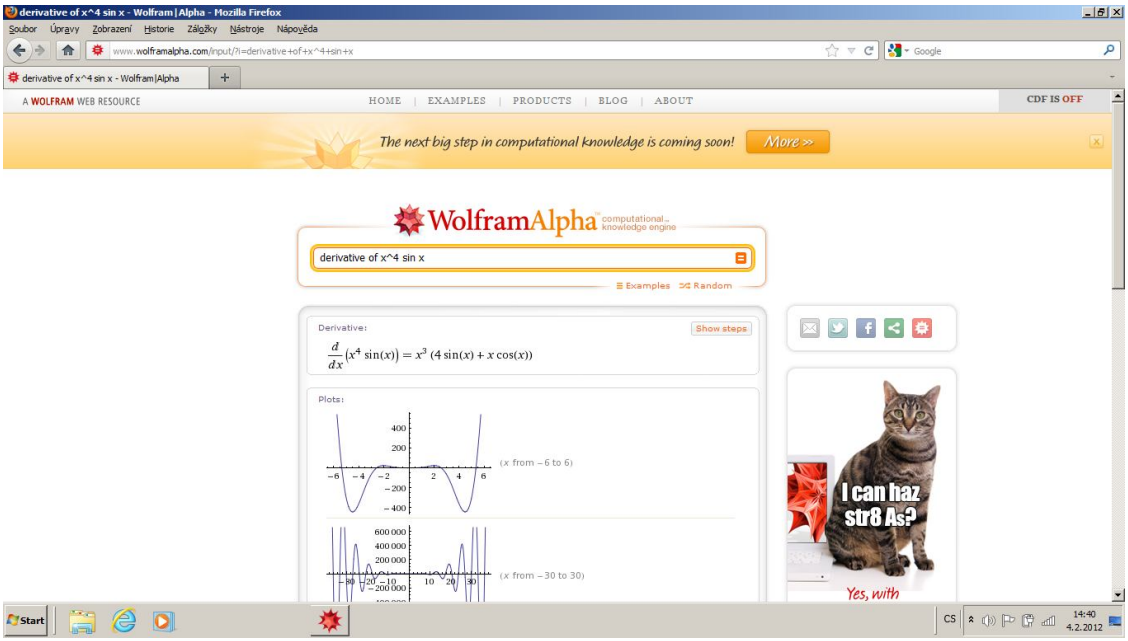
Kromě dynamické nápovědy nabízí tento program ucelenou elektronickou dokumentaci (**Help > Documentation Center**, nebo klávesa F1), přehled nápověd k funkcím (**Help > Function Navigator**) a virtuální příručku (**Help > Virtual Book**). Na internetovém portálu firmy Wolfram Research demonstrations.wolfram.com nabízí řadu hotových aplikací

ke stažení a další zdroje informačních databází. Další užitečné nástroje a výukové soubory od výše jmenované firmy najdeme i na www.wolfram.com/webresources.html.



Obrázek 18 – www.wolfram.com/webresources.html

Velmi zajímavým odkazem na vyzkoušení je wolframalpha.com, kdy můžeme bez nutnosti programu MATHEMATICA vyzkoušet všechny příkazy, funkce a výpočty. Systém spolupracuje s celou řadou databází na internetu. Zde smíme zadat jakýkoli dotaz v angličtině, nebo zapíšeme libovolný pojem v angličtině.



Obrázek 19 – wolframalpha.com – výsledek dotazu derivate of x^4 sin x

Přehled funkcí programu nalezneme na portále functions.wolfram.com.

2 Algebraické výrazy a řešení rovnic

2.1 Algebraické výrazy a proměnné

Algebraické výrazy se skládají nejen z čísel, operací a závorek na určení preferencí, ale také z proměnných, které jsou označené písmeny. Názvy proměnných mohou být libovolné, jedinou podmínkou je, aby začínaly písmenem. Nezapomeňme, že program MATHEMATICA je case-sensitive (rozlišuje malá a velká písmena). Aby se nové názvy nedostaly do konfliktu s již předdefinovanými funkcemi programu, doporučujeme proměnné začít malými písmeny. Potom nehrozí předefinování původních funkcí programu.

Příklad 6

Vytvořte proměnnou *a* hodnoty 5 a vypočtěme hodnotu výrazu $\sqrt{9-a}$.

```
In[3]:= a = 5
        Sqrt[9 - a]

Out[3]=
  5

Out[4]=
  2
```

Obrázek 20 – výstup z příkladu

Na příkladu vidíme, že do jedné vstupní buňky zadáme více výrazů. Každý výraz se přepočítá, proto ve výsledku získáme nejprve dosazenou hodnotu, a pak výsledek druhého výrazu. Pokud výstupy některých výpočtů nechceme zobrazit, stačí zapsat za příkaz středník (viz. následující obrázek).

```
In[5]:= a = 5;
        Sqrt[9 - a]

Out[6]=
  2
```

Obrázek 21 – potlačení výstupu ukončením příkazu středníkem

Někdy je výhodné zadání výrazu a teprve později dosazení proměnné konkrétní hodnotou. V takovém případě s operátorem /. vložíme konkrétní výraz. Za proměnnou nemusíme dosazovat konkrétní čísla, využijme i symbolického výpočtu, kdy za proměnnou dosazujeme jiný výraz.

Příklad 7

Označte *a* výraz $\frac{x^2-1}{x+2}$, dosaďte za *x* hodnotu 3 a *y* + 1. K zápisu použijte nástroje na paletě **Classroom Assistant** v sekci **Calculator** na záložce **Basic**.

```
In[1]:= a = (x^2 - 1)/(x + 2);
        a /. x → 3
        a /. x → y + 1

Out[2]= 8/5

Out[3]= (-1 + (1 + y)^2)/(3 + y)
```

Obrázek 22 – Příklad 7

Příklad 8

Zapište výraz $a = \frac{x^2 - y^2}{x + 2y}$, dosaďte za x a y hodnotu 5 a 3.

```
In[1]:= a = (x^2 - y^2)/(x + 2 y);
        a /. {x → 5, y → 3}

Out[2]= 16/11
```

Obrázek 23 – Příklad 8

Pokud jsme zapomněli definici, nebo hodnotu proměnné (funkce), stačí se dotázat pomocí dotazu *? X*, kde *X* je název proměnné. Ale i pouhé zadání proměnné nám zobrazí aktuální stav. U proměnné, která není definovaná, zahlásí dotaz chybu a při dotazu na hodnotu zobrazí pouze její název. To je výhoda symbolických výpočtů.

```
In[3]:= ? a

Global`a

a = (x^2 - y^2)/(x + 2 y)

In[4]:= a

Out[4]= (x^2 - y^2)/(x + 2 y)

In[5]:= ? z

Information::notfound : Symbol z not found. »

In[6]:= z

Out[6]= z
```

Obrázek 24 – dotaz na proměnnou

Jak jsme už řekli, Kernel si pamatuje všechny proměnné kdekoliv uvedené. Z tohoto důvodu je vhodnější před dalším zadáním, nebo použitím proměnné její uvolnění a zrušení všech předešlých nastavení příkazem Clear[*X*], nebo přiřazením hodnoty *X* = . (*X* je název pro-

měnné). Odstranění všech hodnot a definic provedeme příkazem ClearAll[*X*]. Žádná z uvedených formulí nemá výstup.

```
In[7]:= Clear[a]

In[8]:= a =.
```

Obrázek 25 – uvolnění proměnné

Odstranění všech proměnných provedeme restartováním Kernelu pomocí příkazu z hlavního menu **Evaluation > Quit Kernel > Local**.

2.2 Rovnice

Hledání řešení mnohých úloh vede k sestavení rovnic a jejich soustav. U rovnic hledáme, pro které proměnné se levá strana rovnice rovná pravé straně. Na řešení některých rovnic existují standardní obecné postupy. Pro takové rovnice budeme volit funkci Solve[], vstupní parametry jsou rovnosti (rovnice) a seznam neznámých. Pro numerické výpočty volíme metodu NSolve[]. Složitější rovnice, kde je nutné pro výpočet vycházet z pevného bodu, použijeme metodu FindRoot[]. K řešení nerovnic je vhodnější použít funkci Reduce[]. Program MATHEMATICA hledá vždy řešení v oboru komplexních čísel, obor řešení můžeme tedy omezit parametry Reals, Integers a Complexes.

Příklad 9

Nalezněte řešení rovnice $30 + x^3 = 4x^2 + 11x$, v tomto případě jsou to tři různá řešení.

```
In[1]:= Solve[30 + x^3 == 4 x^2 + 11 x, x]

Out[1]=
{{x → -3}, {x → 2}, {x → 5}}
```

Obrázek 26 – Příklad 9

Příklad 10

Hledejte řešení rovnice $0 = 4x^2 + x + 1$.

```
In[1]:= Solve[0 == 4 x^2 + x + 1, x]

Out[1]=
{{x → 1/8 (-1 - i √15)}, {x → 1/8 (-1 + i √15)}}
```

Obrázek 27 – Příklad 10 – řešení v oboru komplexních čísel

Pokud chceme pouze reálná řešení, omezme oblast řešení klíčovým slovem Reals.

```
In[2]:= Solve[0 == 4 x^2 + x + 1, x, Reals]

Out[2]=
{}
```

Obrázek 28 – Příklad 10 – řešení v oboru reálných čísel

Prázdná závorka značí, že množina řešení je prázdná (nemá řešení).

Příklad 11

Najděte řešení rovnice $1 = \dfrac{2x - (x-1)}{x+1}$.

```
In[1]:= Solve[1 == (2 x - (x - 1))/(x + 1), x]

Out[1]=
{{}}
```

Obrázek 29 – Příklad 11 – Solve[]

Dvojité složené závorky znamenají, že řešením je interval z definičního oboru. Pro potvrzení, že je to celý interval, použijme funkci Reduce[]. Ta hledá interval v definičním oboru, který splňuje danou podmínku.

```
In[1]:= Reduce[1 == (2 x - (x - 1))/(x + 1), x]

Out[1]=
True
```

Obrázek 30 – Příklad 11 – Reduce[]

Odpověď True znamená, že řešením je celý definiční obor.

Příklad 12

Najděte řešení rovnice $|x+1|+|x-2|=3$.

```
In[1]:= Solve[Abs[x + 1] + Abs[x - 2] == 3, x, Reals]

Solve::fulldim : The solution set contains a full-dimensional component; use Reduce for complete solution information. >>

Out[1]=
{{}}
```

Obrázek 31 – Příklad 12 – Solve[]

Chybové hlášení nás upozorňuje, že je vhodné zvolit jinou metodu výpočtu. Dvojité složené závorky opět oznamují, že řešením je interval z definičního oboru.

```
In[1]:= Reduce[Abs[x + 1] + Abs[x - 2] == 3, x, Reals]

Out[1]=
-1 ≤ x ≤ 2
```

Obrázek 32 – Příklad 12 – Reduce[]

V tomto případě dostáváme jako řešení interval. Metoda Reduce[] je především určena na hledání řešení nerovnic.

Příklad 13

Najděte řešení rovnice $210+187x-44x^3+x^7=0$.

```
In[1]:= Solve[210 + 187 x - 44 x^3 + x^7 == 0, x]

Out[1]=
{{x → Root[210 + 187 #1 - 44 #1^3 + #1^7 &, 1]},
 {x → Root[210 + 187 #1 - 44 #1^3 + #1^7 &, 2]}, {x → Root[210 + 187 #1 - 44 #1^3 + #1^7 &, 3]},
 {x → Root[210 + 187 #1 - 44 #1^3 + #1^7 &, 4]}, {x → Root[210 + 187 #1 - 44 #1^3 + #1^7 &, 5]},
 {x → Root[210 + 187 #1 - 44 #1^3 + #1^7 &, 6]}, {x → Root[210 + 187 #1 - 44 #1^3 + #1^7 &, 7]}}
```

Obrázek 33 – Příklad 13 – Solve[]

Metoda funkce Solve[] nedokáže vypočítat kořeny této rovnice. Proto užijeme numerické metody NSolve[].

```
In[2]:= NSolve[210 + 187 x - 44 x^3 + x^7 == 0, x]

Out[2]=
  {{x → -2.24648}, {x → -1.23544 - 0.665006 i}, {x → -1.23544 + 0.665006 i},
   {x → 0.0803462 - 2.86879 i}, {x → 0.0803462 + 2.86879 i},
   {x → 2.27833 - 0.758072 i}, {x → 2.27833 + 0.758072 i}}
```

Obrázek 34 – Příklad 13 – NSolve[]

Podařilo se nám najít všech sedm řešení v oboru komplexních čísel. Pokud i tato metoda selže, můžeme se pokusit najít nejbližší kořen od zadaného bodu metodou FindRoot[]. Jako výchozí bod si zvolme -3.

```
In[3]:= FindRoot[210 + 187 x - 44 x^3 + x^7 == 0, {x, -3}]

Out[3]=
  {x → -2.24648}
```

Obrázek 35 – Příklad 13 – FindRoot[]

Tato metoda najde pouze jedno řešení, dokonce v mnoha případech nemusí řešení nalézt, i když existuje.

Příklad 14

Najděte řešení soustavy rovnic $3(x-2)+2y=x+y$, $4x+5(y+x)=3x-6$.

```
In[1]:= Solve[{3 (x - 2) + 2 y == x + y, 4 x + 5 (y + x) == 3 x - 6}, {x, y}]

Out[1]=
  {{x → 9, y → -12}}
```

Obrázek 36 – Příklad 14

Příklad 15

Nalezněte řešení soustavy rovnic $3(x-2)+2y=5+x$, $3(x-2)+2y=-6+x$.

```
In[1]:= Solve[{3 (x - 2) + 2 y == 5 + x, 3 (x - 2) + 2 y == -6 + x}, {x, y}]

Out[1]=
  {}
```

Obrázek 37 – Příklad 15

V tomto případě soustava nemá řešení.

Příklad 16

Vypočítejte řešení soustavy rovnic $2x-3y=5$, $4x-6y=10$.

```
In[1]:= Solve[{2 x - 3 y == 5, 4 x - 6 y == 10}, {x, y}]

  Solve::svars : Equations may not give solutions for all "solve" variables. >>

Out[1]=
  {{y → -5/3 + (2 x)/3}}
```

Obrázek 38 – Příklad 16

V tomto případě dostaneme nekonečně mnoho řešení; $x \in \mathbf{R}$, y je dána předpisem s parametrem x, $y = -\frac{5}{3} + \frac{2x}{3}$.

Příklad 17

Nalezněte řešení těchto nerovnic:

a) $x^2 + 5x - 6 \geq 0$.

```
In[1]:= Reduce[x^2 + 5 x - 6 >= 0, x]

Out[1]=
x ≤ -6 || x ≥ 1
```

Obrázek 39 – Příklad 17 – zadání a)

Symbol | | představuje spojku nebo (matematický zápis $\vee$), symbol && představuje spojku a (zároveň v matematickém zápisu $\wedge$). Klasický zápis získáme funkcí TraditionalForm[].

```
Reduce[x^2 + 5 x - 6 ≥ 0, x] // TraditionalForm

Out[1]//TraditionalForm=
x ≤ -6 ∨ x ≥ 1
```

nebo :

```
In[2]:= TraditionalForm[Reduce[x^2 + 5 x - 6 ≥ 0, x]]

Out[2]//TraditionalForm=
x ≤ -6 ∨ x ≥ 1
```

Obrázek 40 – Příklad 17 – zadání a) TraditionalForm[]

Znak ≤ a ≥ na klávesnici nenajdeme, lze je ale nahradit kombinací kláves <= a >=. Pokud chceme použít přímo tyto speciální symboly, vyhledáme je v hlavní nabídce programu **Palettes > Classroom Assistant** na paletě **Typesetting**.

b) $x^2 + 2x + 4 \leq 0$.

```
In[1]:= Reduce[x^2 + 2 x + 4 ≤ 0, x]

Out[1]=
False
```

Obrázek 41 – Příklad 17 – zadání b)

Odpověď False znamená, že úloha nemá řešení.

c) $x^2 + 4x + 4 \leq 0$.

```
In[1]:= Reduce[x^2 + 4 x + 4 ≤ 0, x]

Out[1]=
x == -2
```

Obrázek 42 – Příklad 17 – zadání c)

V tomto případě je řešením pouze jeden bod.

Příklad 18

Najděte řešení soustavy nerovnic $3y+2x<8,\ 2y+x\geq 0$.

```
In[5]:= Reduce[{3 y + 2 x < 8, 2 y + x ≥ 0}, {x, y}] // TraditionalForm
```

Out[5]//TraditionalForm=

$$x<16 \bigwedge -\frac{x}{2}\leq y<\frac{1}{3}(8-2x)$$

Obrázek 43 –Příklad 18

3 Práce s grafy

Silným nástrojem s širokou podporou je grafický výstup v programu MATHEMATCA. Program umožňuje kromě jiného zobrazit různé druhy grafů. Ukážeme si použití příkazů Plot[], ListPlot[], ParametricPlot[], PolarPlot[], StreamPlot[], Plot3D[] a Show[] k zobrazení grafů.

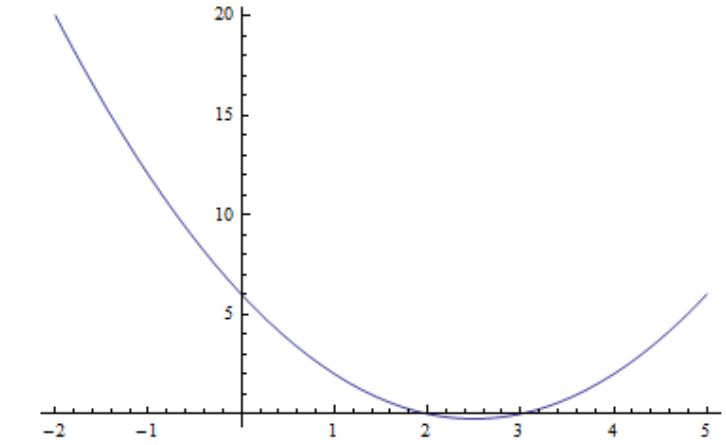
Příkaz Plot[] má dva parametry. Prvním je výraz předpisující funkční hodnotu (seznam funkcí) a druhým je označení proměnné ve výrazu a její rozsah zobrazení: Plot[výraz, {x, x_{min}, x_{max}}].

Příklad 19

Zobrazte graf funkce $x^2 - 5x + 6$ v intervalu od -2 až do 5.

```
In[1]:= Plot[x^2 - 5 x + 6, {x, -2, 5}]
```

Out[1]=



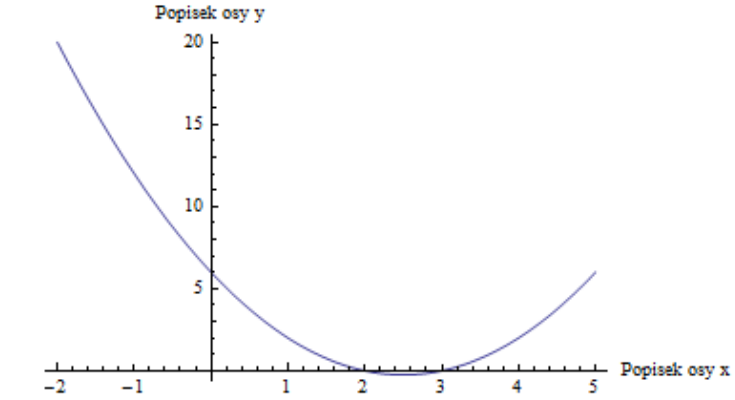
Obrázek 44 – Příklad 19 – zobrazení funkce

U grafu můžeme nastavovat spousty parametrů, kterými upravíme vzhled grafu. Ukážeme si některé zajímavé úpravy.

Doplňme graf o popisek osy x a popisek osy y direktivou AxesLabel.

```
In[2]:= Plot[x^2 - 5 x + 6, {x, -2, 5}, AxesLabel → {"Popisek osy x", "Popisek osy y"}]
```

Out[2]=

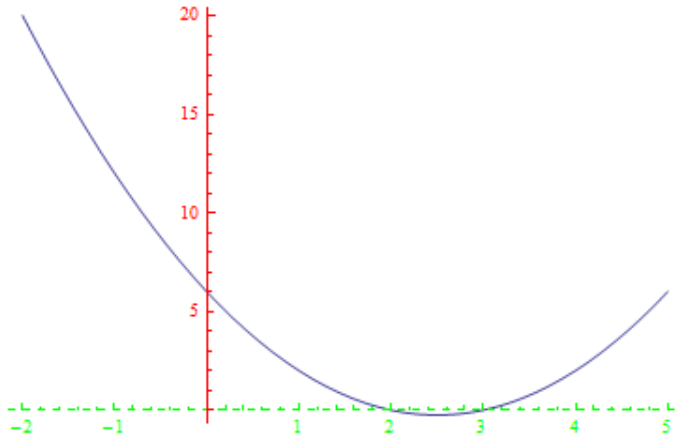


Obrázek 45 – Příklad 19 – popisky os

Upravíme styl osy x na zelenou (Green) zakreslenou přerušovanou čarou (Dashed) a osu y zobrazíme červeně (Red) direktivou AxesStyle.

```
In[3]:= Plot[x^2 - 5 x + 6, {x, -2, 5}, AxesStyle → {Directive[Dashed, Green], Red}]
```

Out[3]=

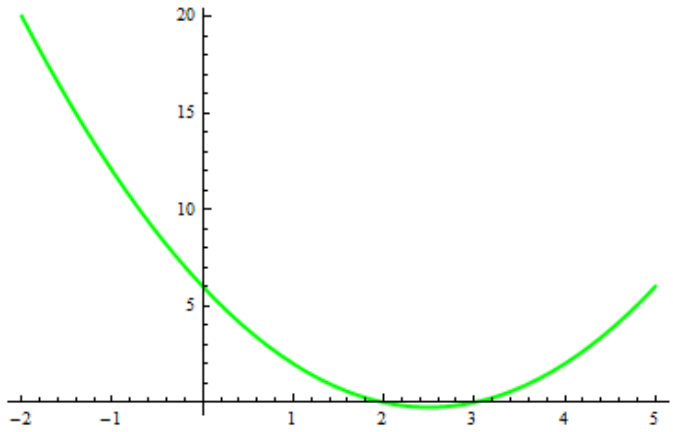


Obrázek 46 – Příklad 19 – zobrazení os

Nastavíme barvu grafu na zelenou (Green), křivku grafu zobrazíme zesílenou (Thick) pomocí direktivy PlotStyle.

```
In[4]:= Plot[x^2 - 5 x + 6, {x, -2, 5}, PlotStyle → {Directive[Thick, Green]}]
```

Out[4]=

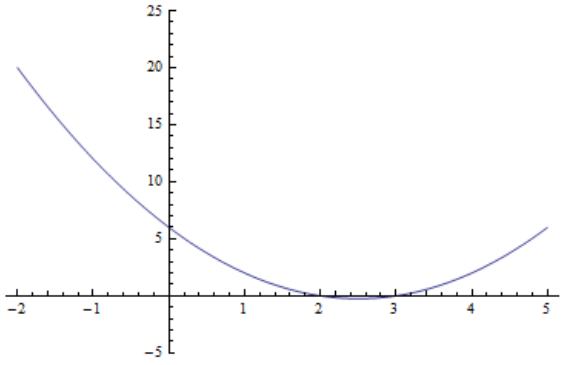


Obrázek 47 – Příklad 19 – styl vykreslení grafu

Direktivou PlotRange rozšíříme zobrazenou oblast grafu ve směru osy *y* na rozsah od -5 do 25.

```
(Debug) In[5]:= Plot[x^2 - 5 x + 6, {x, -2, 5}, PlotRange → {-5, 25}]
```

(Debug) Out[5]=

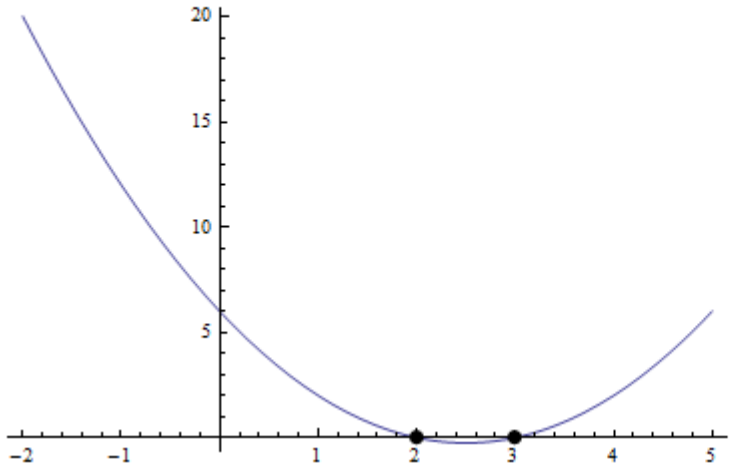


Obrázek 48 – Příklad 19 – nastavení rozsahu osy y

Zvýrazníme průsečíky s osou *x* direktivou Epilog. Použijeme funkci Point[] a PointSize[].

```
In[6]:= Plot[x^2 - 5 x + 6, {x, -2, 5}, Epilog → {PointSize[0.02], Point[{2, 0}], Point[{3, 0}]}]
```

Out[6]=



Obrázek 49 – Příklad 19 – zvýraznění bodů grafu

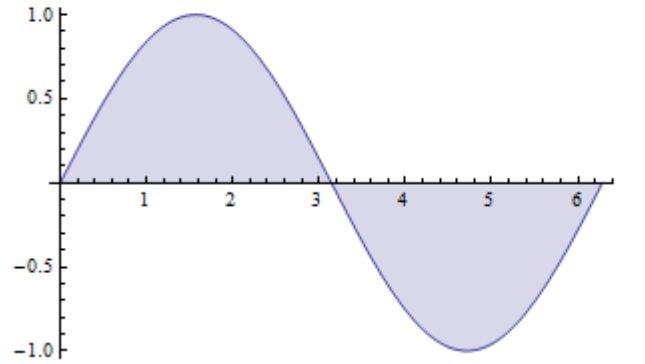
Další zajímavou direktivou je Filling. Ta umožňuje vykreslení plochy mezi grafem funkce a osou *x*, nebo vykreslení plochy mezi grafy dvou funkcí. Celou dokumentaci k této direktivě nalezneme na stránkách helpu. Možnosti této direktivy jsou rozsáhlé a dají se využít nejen v kapitolách o integrálním počtu, ale i v technické dokumentaci, nebo ve fyzice.

Příklad 20

Zvýrazněte plochu mezi grafem a osou *x* u funkce sin *x*.

```
In[1]:= Plot[{Sin[x]}, {x, 0, 2 Pi}, Filling -> Axis]
```

Out[1]=



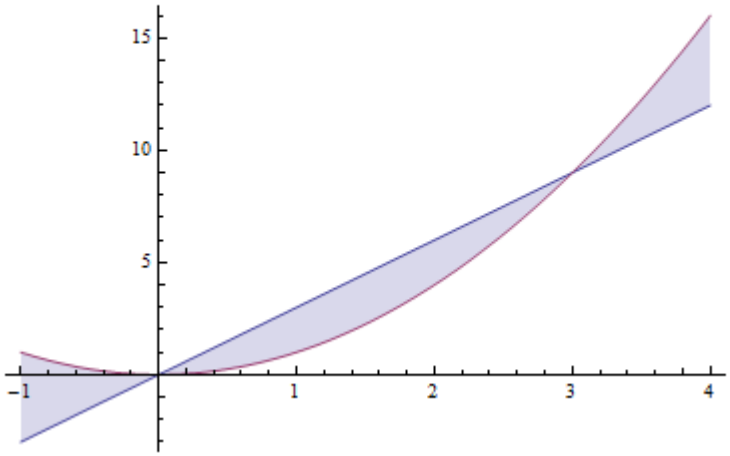
Obrázek 50 – Příklad 20

Pokud chceme zobrazit více funkcí, musíme je uvést jako seznam ve složených závorkách.

Příklad 21

Zvýrazněte plochu vymezenou křivkami $y_1 = 3x$, $y_2 = x^2$.

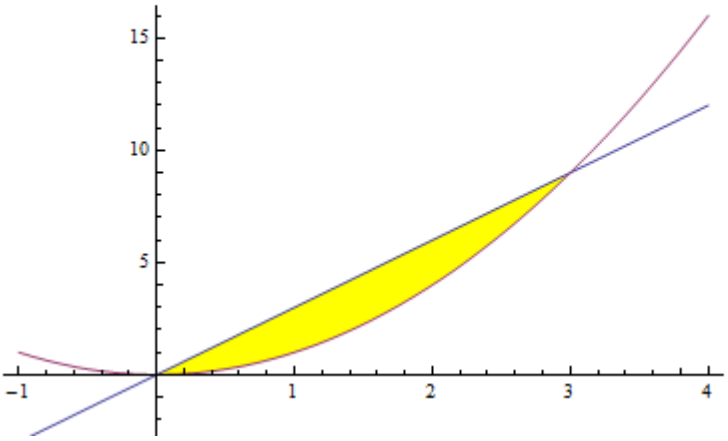
```
In[1]:= Plot[{3 x, x^2}, {x, -1, 4}, Filling → {1 → {2}}]
Out[1]=
```



Obrázek 51 – Příklad 21 – plocha mezi křivkami

Pokud chceme zobrazit určitou část kde $y_1 > y_2$, upravíme příkaz následovně.

```
In[2]:= Plot[{3 x, x^2}, {x, -1, 4}, Filling → {1 → {{2}, {White, Yellow}}}]
Out[2]=
```



Obrázek 52 – Příklad 21 – plocha dané vlastnosti

V této ukázce barva ploch vymezených křivkami není dána vlastnostmi funkcí, je to jen pravidelné střídání barev z definovaného seznamu.

V mnoha případech není funkce zadána klasicky, může být zadána parametricky. V takové situaci použijeme funkce ParametricPlot[]. V prvním parametru je seznam funkcí určující souřadnice, druhý zadává název parametru a jeho rozsah.

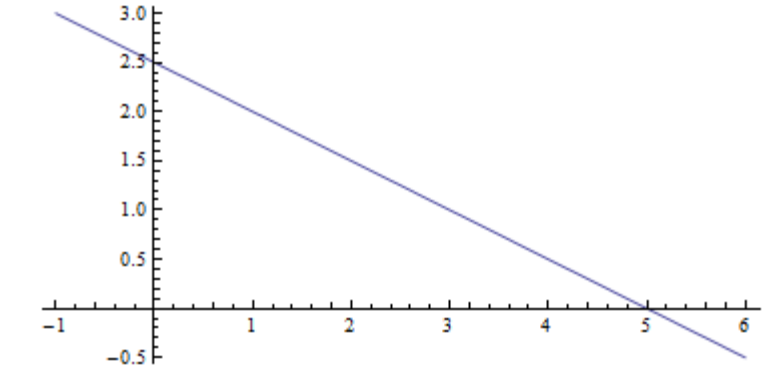
ParametricPlot[$\{x(t), y(t)\}, \{t, t_{min}, t_{max}\}$]

Příklad 22

Zobrazte parametricky zadanou přímku $\{x = 3 + 2t,\ y = 1 - t\}$ s rozsahem parametru od -2 do 1,5.

```
In[1]:=
ParametricPlot[{3 + 2 t, 1 - t}, {t, -2, +1.5}]
```

Out[1]=



Obrázek 53 – Příklad 22

Další možností zadání funkce je definice v polárních souřadnicích. V takovému případě je tu funkce PolarPlot[]. Prvním parametrem je vzorec pro výpočet vzdálenosti *r* od počátku, druhý je označení proměnné pro úhel a vymezení jeho rozsahu.

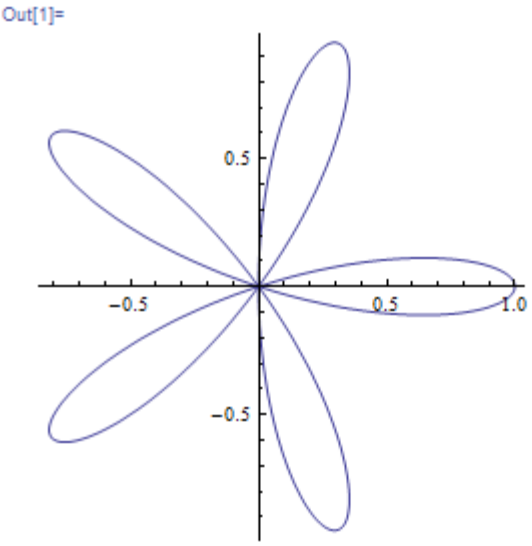
PolarPlot[$r(\varphi)$, $\{\varphi, \varphi_{min}, \varphi_{max}\}$]

Příklad 23

Zobrazte graf polárně zadané funkce $r = \cos(7\phi)$ pro ϕ od 0 do π (pokuste se zapsat ϕ posloupností kláves Esc – f – Esc, takto se dá zapsat celá řecká abeceda).

```
In[1]:= PolarPlot[Cos[5 ϕ], {ϕ, 0, π}]
```

Out[1]=



Obrázek 54 – Příklad 23

Významným typem grafů je graf vektorového pole. K vykreslení tohoto grafu slouží funkce StreamPlot[], nebo VectorPlot[]. V prvním parametru zadáváme souřadnice vektoru, v druhém název proměnné na ose *x* s jejím rozsahem a v třetím název a rozsah proměnné na ose *y*.

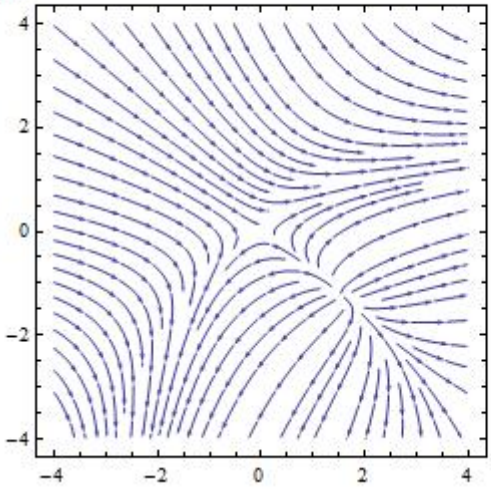
StreamPlot[$\{v_x(x, y), v_y(x, y)\}, \{x, x_{min}, x_{max}\}, \{y, y_{min}, y_{max}\}$]

Příklad 24

Zobrazte vektorové pole ($v_x = x^2 + 2y$, $v_y = x - y^2$) pro x a y v rozsahu od -4 do 4.

```
In[1]:= StreamPlot[{x^2 + 2 y, x - y^2}, {x, -4, 4}, {y, -4, 4}]
```

Out[1]=



Obrázek 55 – Příklad 24

V některých příkladech může být graf zadán jen seznamem několika bodů, pak použijeme funkce ListPlot[]. Parametry této funkce jsou hodnoty y, nebo přímo uspořádané dvojice představující konkrétní souřadnice grafu.

ListPlot[{y_1, y_2, ...}]

nebo

ListPlot[{{x_1, y_1},{x_2, y_2}, ...}].

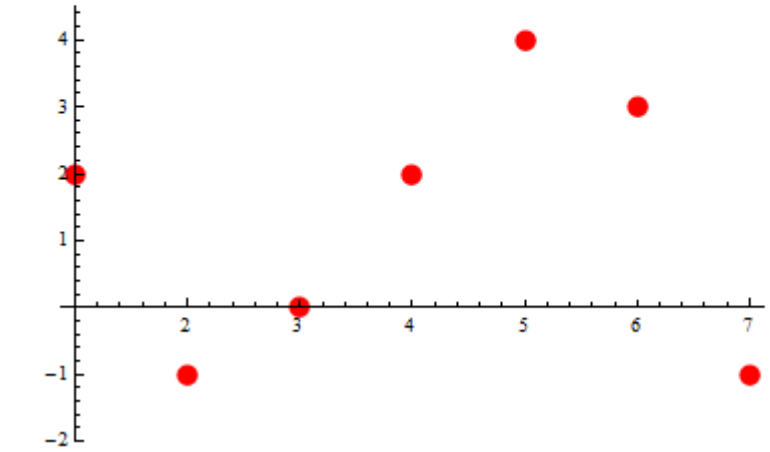
Příklad 25

Zobrazte graf funkce zadané tabulkou. Body zobrazíme červeně s velikostí 0,03 a rozšíříme rozsah osy y na interval od -2 do 4,5.

x	1	2	3	4	5	6	7
y	2	-1	0	2	4	3	-1

```
In[1]:= ListPlot[{{1, 2}, {2, -1}, {3, 0}, {4, 2}, {5, 4}, {6, 3}, {7, -1}},
        PlotStyle → {Red, PointSize[0.03]}, PlotRange → {-2, 4.5}]
```

Out[1]=



Obrázek 56 – Příklad 25

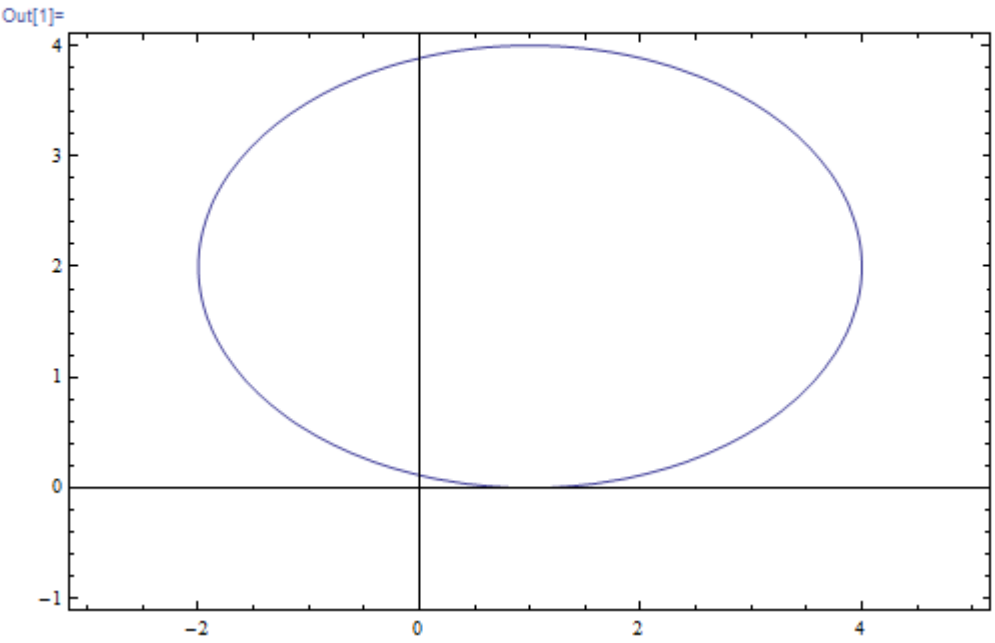
Další skupinou 2D grafů jsou grafy kuželoseček a implicitně zadané funkce.

Příklad 26

Zobrazte graficky kuželosečku $4x^2 + 9y^2 - 8x - 36y + 4 = 0$.

```
In[1]:= ContourPlot[4 x^2 + 9 y^2 - 8 x - 36 y + 4 == 0, {x, -3, 5}, {y, -1, 4},
        Axes → True, AspectRatio → Automatic]
```

Out[1]=



Obrázek 57 – Příklad 26

Direktiva Axes nám umožní znázornit osy a direktiva AspectRati nastavuje stejné měřítko na obou osách.

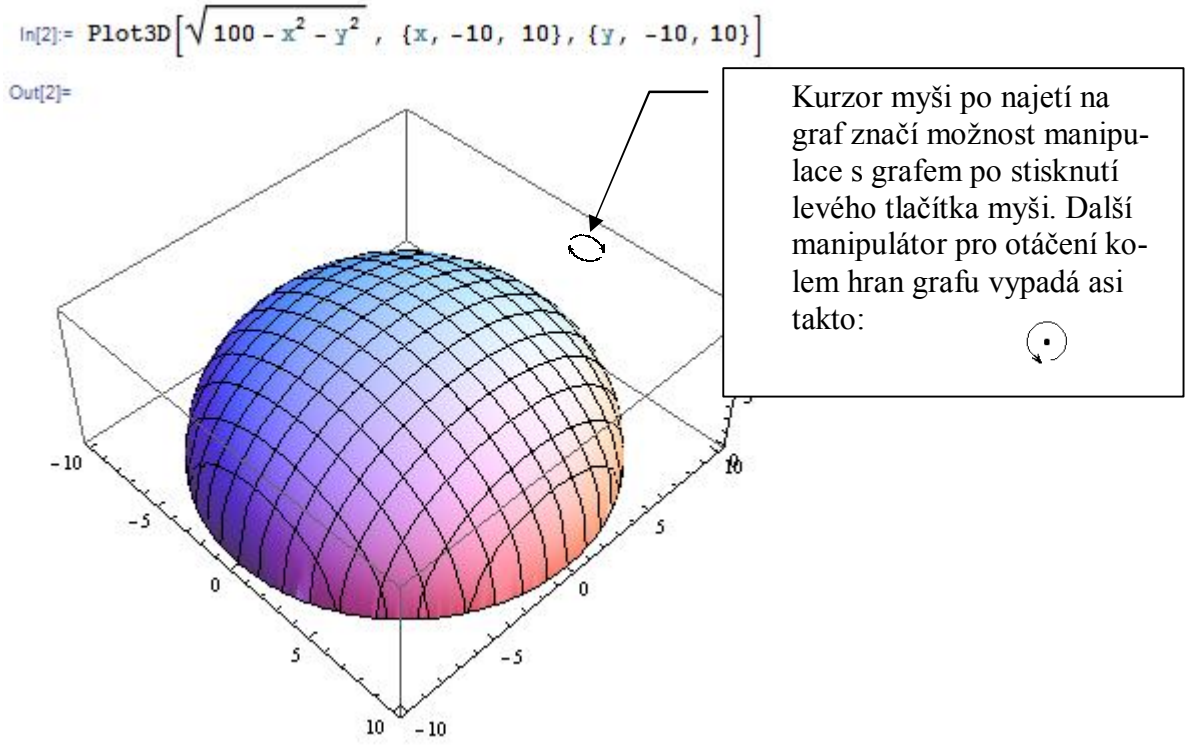
Samostatnou kapitolou mezi grafy jsou 3D grafy. Ukážeme si tři typy těchto grafů vytvořené funkcemi Plot3D[], VectorPlot3D[], ParametricPlot3D[]. Všechny tyto grafy můžeme pomocí držení levého tlačítka myši otáčet nebo jinak s nimi manipulovat.

U funkce Plot3D[] musíme zadat výraz pro výpočet třetí souřadnice, a pak rozsah souřadnice x a y.

Plot3D[$f(x,y)$, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}]

Příklad 27

Zobrazte graf funkce $z(x,y) = \sqrt{100 - x^2 - y^2}$ v rozsahu x a y od -10 do 10.



Obrázek 58 – Příklad 27

Funkce ParametricPlot3D[] má opět dva parametry. První představuje seznam pro výpočet souřadnic x, y, z a druhý představuje označení parametru a jeho rozsah.

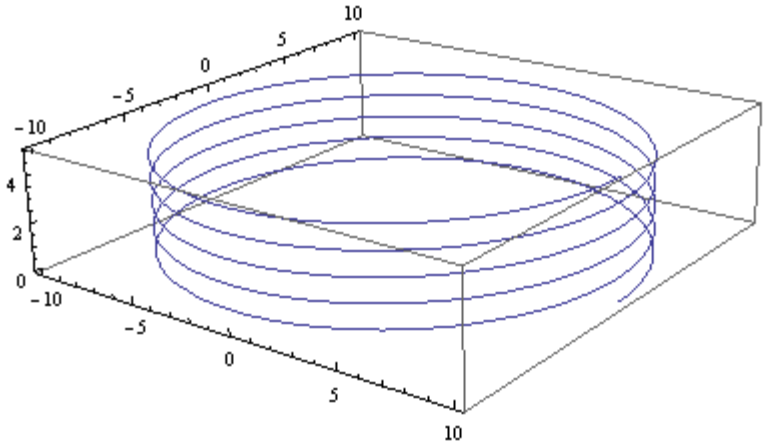
ParametricPlot3D[{$x(t)$, $y(t)$, $z(t)$}, {t, t_{min}, t_{max}}]

Příklad 28

Zobrazte graf šroubovice zadané parametricky $x(t) = 10\cos t$, $y(t) = 10\sin t$, $z(t) = \frac{t}{2\pi}$.

Průběh parametru zadejte od 0 do 10π.

```
In[2]:= ParametricPlot3D[{10 Cos[t], 10 Sin[t], t / (2 π)}, {t, 0, 10 π}]
Out[2]=
```



Obrázek 59 – Příklad 28

Poslední ukázkou 3D grafu je vektorový graf. Funkce VectorPlot3D[] má čtyři parametry. První určuje velikost vektoru a zbylé určují označení souřadnic a jejich rozsah.

VectorPlot3D[$\{v_x(x,y,z), v_y(x,y,z), v_z(x,y,z)\}$,$\{x, x_{min}, x_{max}\}$, $\{y, y_{min}, y_{max}\}$, $\{z, z_{min}, z_{max}\}$]

Příklad 29

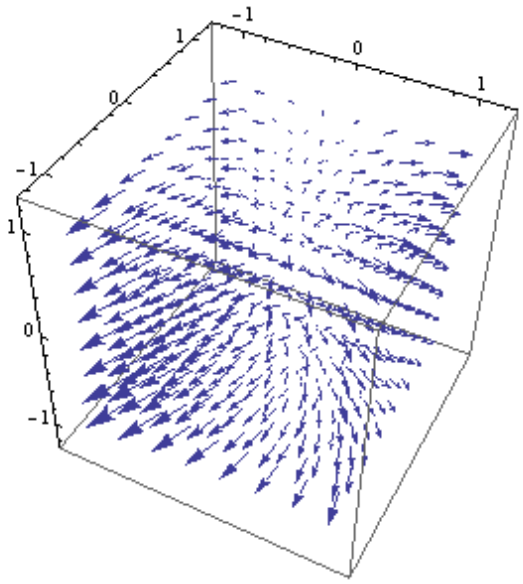
Zobrazte graf vektorového prostoru v rozsahu souřadnic od -1 do 1. Vektorový prostor:

$v_x(x,y,z) = 2x + z - y^2;$

$v_y(x,y,z) = x + 2y - z^2;$

$v_z(x,y,z) = -x^2 - y^2 + z^2.$

```
In[1]:= VectorPlot3D[{2 x + z - y^2, x + 2 y - z^2, -x^2 - y^2 + z}, {x, -1, 1}, {y, -1, 1},
        {z, -1, 1}]
Out[1]=
```



Obrázek 60 – Příklad 29

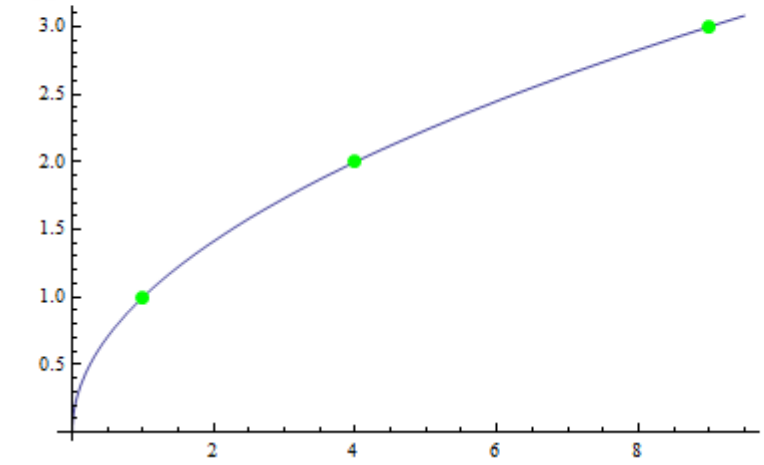
Dva různé grafy, nebo grafické objekty spojíme do jednoho objektu pomocí funkce Show[]. Její parametry jsou objekty, které chceme spojit do jednoho výstupu.

Příklad 30

a) Zobrazte graf funkce $\sqrt{x}$ a bodový graf pro body [1, 1], [4, 2], [9, 3].

```
In[1]:= Show[Plot[√x , {x, 0, 9.5}],
          ListPlot[{{1, 1}, {4, 2}, {9, 3}}, PlotStyle → {Green, PointSize[0.02]}]]
```

Out[1]=



Obrázek 61 – Příklad 30 – zadání a)

b) Zobrazte zkopírovaný obrázek ze schránky společně s křivkou grafu $\sqrt{100x}$.

```
In[2]:= Show[ , Plot[√100 x , {x, 0, 100} , PlotStyle → Blue]]
```

Out[2]=

Obrázek 62 – Příklad 30 – zadání b)

Z poslední ukázky plyne, že můžeme spojovat libovolné grafické objekty.

V nápovědě si prohlédneme i další možnosti práce s grafy. Rozbor všech možností by vydal na samostatnou knihu.

Ke grafům často patří tabulky. Ty mohou být doplňkem, nebo přímo zdrojem grafu. Výstupem funkce Table[] bývá pole. Prvním parametrem je předpis výrazu, druhým parametrem je seznam, kde je určená proměnná, počáteční hodnota proměnné, koncová hodnota proměnné a krok.

Table[předpis pro výpočet hodnoty,{parametr, od, do, krok}]

Příklad 31

Vytvořte tabulku hodnot funkce zadané výrazem $x^2 - x$ pro $x = -2, -1, 0, 1, 2$.

```
In[1]:= Table[x^2 - x, {x, -2, 2, 1}]

Out[1]=
{6, 2, 0, 0, 2}

In[2]:= Table[x^2 - x, {x, -2, 2}]

Out[2]=
{6, 2, 0, 0, 2}
```

Obrázek 63 – Příklad 31

Pokud je krok roven jedné, nemusíme ho uvádět. Takto získaný seznam je bez hodnot proměnné. Stačí tedy uvést, že chceme vytvořit uspořádané dvojice s hodnotou *x* a výslednou funkční hodnotou.

```
In[3]:= Table[{x, x^2 - x}, {x, -2, 2}]

Out[3]=
{{-2, 6}, {-1, 2}, {0, 0}, {1, 0}, {2, 2}}
```

Obrázek 64 – Příklad 31 – uspořádané dvojice hodnot

Takové hodnoty uspořádáme do grafické tabulky. Ukážeme si teď několik možných tabulek pomocí příkazu TableForm[]. První variantou je sloupcová tabulka se zarovnáním doprava (Right) direktivou TableAlignments a druhá představuje řádkovou variantu tabulky (Row) pomocí direktivy TableDirections.

```
In[4]:= TableForm[Table[{x, x^2 - x}, {x, -2, 2}], TableAlignments → Right]

Out[4]//TableForm=
-2   6
-1   2
 0   0
 1   0
 2   2

In[5]:= TableForm[Table[{x, x^2 - x}, {x, -2, 2}], TableAlignments → Right,
          TableDirections → Row]

Out[5]//TableForm=
-2  -1  0  1  2

 6   2  0  0  2
```

Obrázek 65 – Příklad 31 – řádková a sloupcová tabulka

Pro lepší přehlednost lze doplnit údaje o hlavičku pomocí direktivy TableHeadings.

```
In[6]:= TableForm[Table[{x, x^2 - x}, {x, -2, 2}], TableAlignments → Right,
          TableHeadings → {None, {"x", "f(x)"}} ]

Out[6]//TableForm=
 x   f(x)
-2     6
-1     2
 0     0
 1     0
 2     2
```

Obrázek 66 – Příklad 31 – hlavička tabulky

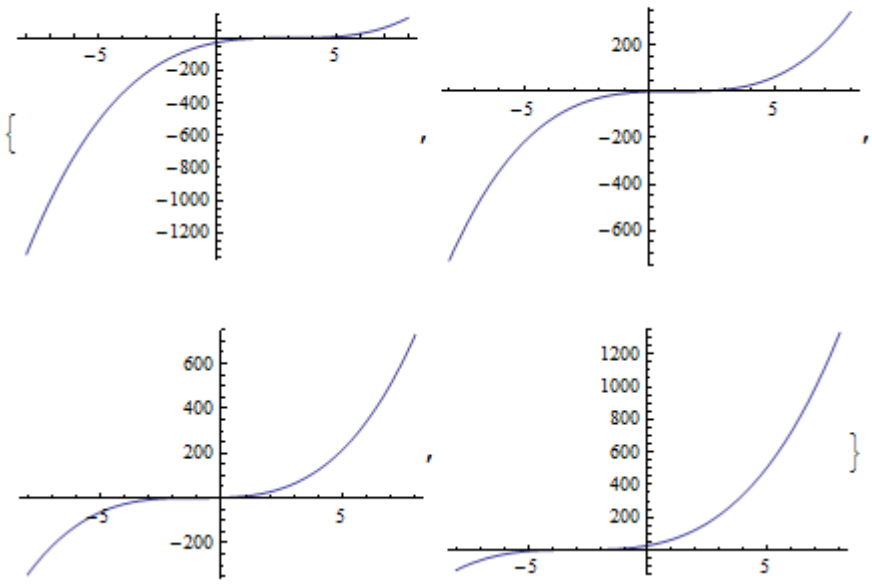
Další možností je použít příkaz Table[] na vykreslení tabulky grafů. To je výhodné pro zobrazení funkcí s parametrem.

Příklad 32

Vytvořte grafy funkcí $(x+c)^3$ pro hodnoty parametru c -3, -1, 1, 3.

```
In[1]:= Table[Plot[(x + c)^3, {x, -8, 8}], {c, -3, 3, 2}]
```

Out[1]=



Obrázek 67 – Příklad 32

Na obrázku krásně vidíme posunutí grafu funkce podle parametru c. Při zvyšování hodnoty c se graf posouvá doleva.

4 Funkce

Tato kapitola vás uvede do dalších možností využití programu MATHEMATICA na zavádění a vyšetření základních vlastností funkcí. Funkce je zobrazením z množiny R do množiny R. Funkci ve zjednodušené podobě můžeme vnímat jako předpis přiřazení k daným hodnotám $x \in$ Df právě jedné hodnoty y $\in$ Hf. Df nazýváme definiční obor funkce a Hf nazýváme obor hodnot funkce. Předpis můžeme zapsat:

f [*x*_, *y*_]= výraz;

f	je název funkce;
x, *y*	jsou proměnné funkce, aby počítač rozpoznal, kde funkci definujete a kde vypočítáváme její hodnotu, značí se v definici proměnné s podtržítkem.

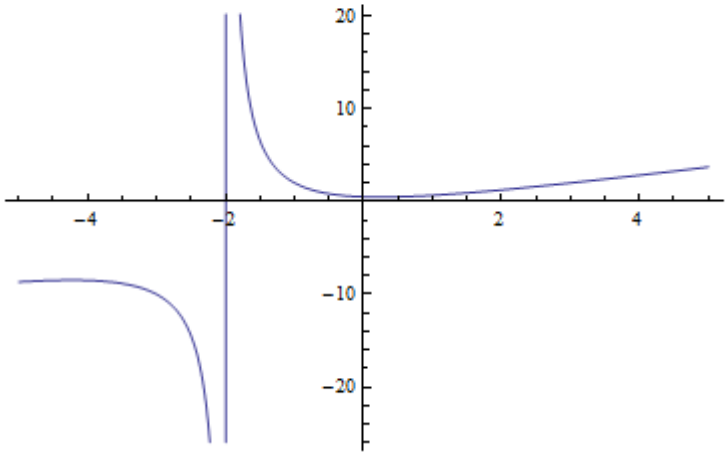
Příklad 33

Zaveďte funkci $f(x)=\frac{1+x^2}{x+2}$, zobrazte hodnoty funkce $f(5), f(8), f(x+1)$ a graf této funkce.

```
In[1]:= f[x_] = (1 + x^2)/(x + 2)

Out[1]=
        1 + x^2
        -------
        2 + x

In[2]:= f[5]

Out[2]=
        26
        --
        7

In[3]:= f[8]

Out[3]=
        13
        --
        2

In[4]:= f[x + 1]

Out[4]=
        1 + (1 + x)^2
        -------------
            3 + x
```

Obrázek 68 – Příklad 33 – funkční hodnoty

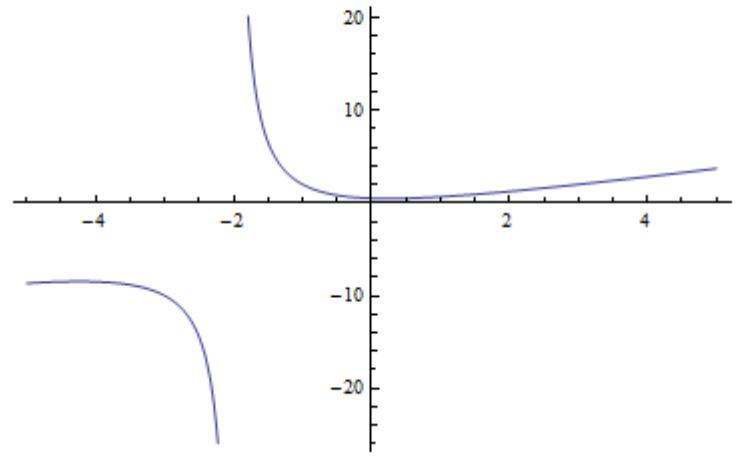
```
In[5]:= Plot[f[x], {x, -5, 5}]
Out[5]=
```



Obrázek 69 – Příklad 33 – první graf

V tomto grafu dochází ke zkreslení. Funkce v bodě 2 není definovaná. Pro odstranění tohoto zkreslení použijeme direktivu Exclusions, které udáme, kde funkce není definovaná. V našem případě je to pro hodnotu $x = 2$, nebo to provedeme širší definicí, funkce není definována v bodě, kde jmenovatel funkce nabývá hodnoty 0.

```
In[6]:= Plot[f[x], {x, -5, 5}, Exclusions → {x + 2 == 0}]
Out[6]=
```



Obrázek 70 – Příklad 33 – vylepšený graf

Protože Kernel si pamatuje všechny zadané proměnné i funkce, je vhodné před použitím danou proměnnou z paměti odstranit příkazem Clear[]. Pokud si nevzpomeneme na definici funkce, stačí zadat otazník před jméno funkce bez parametrů, jak jsme si ukázali v kapitole nápověda.

```
In[7]:= ? f

Global`f

f[x_] = (1+x^2)/(2+x)

In[8]:= Clear[f]

In[9]:= ? f

Global`f
```

Obrázek 71 – nápověda k funkci a odstranění funkce

Ukázali jsme si, jak definovat funkce jedním výrazem. Taková definice se nazývá přímou definicí funkce. Pokud předpis funkce záleží na hodnotě proměnné, označuje se taková definice funkce za odloženou. Program MATHEMATICA teprve po zadání proměnné vybere daný předpis pro funkci. Taková funkce se definuje pomocí := a za znaky /; se zapisují podmínky platnosti definice.

Příklad 34

Zadejte funkci formou odložené definice $f(x)=\begin{cases} x & \text{pro} \quad x \le 0. \\ 1+\sin(x) & \text{pro} \quad x > 0. \end{cases}$

```
In[1]:= Clear[f]

In[2]:= f[x_] := 1 - x /; x ≤ 0
        f[x_] := 1 + Sin[x] /; x > 0
```

Obrázek 72 – Příklad 34 – první výstup

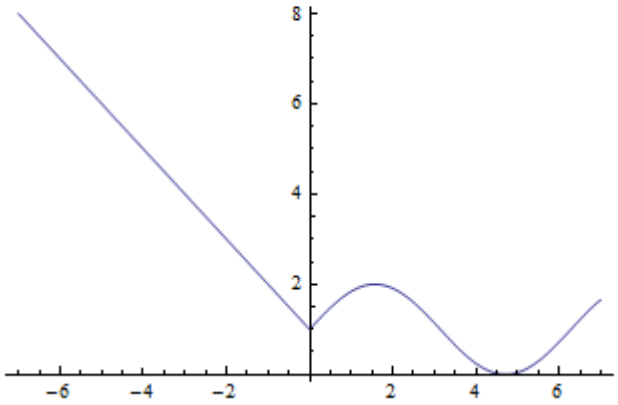
Prvním příkazem vymažeme dřívější definice funkce *f* a druhým nadefinujeme novou funkci. Tyto příkazy nemají žádný výstup. Proto pokud chceme vidět celý předpis funkce, použijeme příkazu ? *f*. Potvrzením je i výpočet konkrétních funkčních hodnot.

```
In[4]:= ? f

Global`f

f[x_] := 1 - x /; x ≤ 0

f[x_] := 1 + Sin[x] /; x > 0

In[5]:= f[-1]

Out[5]=
2

In[6]:= f[π/2]

Out[6]=
2
```

Obrázek 73 – Příklad 34 – informace o funkci a funkční hodnoty

Takovou funkci můžeme i zobrazit.

```
In[7]:= Plot[f[x], {x, -7, 7}]
Out[7]=
```



Obrázek 74 – Příklad 34 – graf

Často nás zajímají různé vlastnosti funkcí. Jednou z nejdůležitějších vlastností je Df definiční obor funkce. Aplikace nemá speciální funkci pro určení definičního oboru. Vždy musíme omezení definičního oboru určit sami. Připomeňme si základní pravidla.

1) Jmenovatel zlomků musí být různý od 0.

2) Výraz pod sudou odmocninou musí být nezáporný.

3) Výraz v logaritmu musí být kladný.

4) Výraz ve funkci tg (x) se nesmí rovnat $\frac{\pi}{2}+k\pi; k \in \mathbf{Z}$.

5) Výraz ve funkci arcsin (x) a arccos (x) musí splňovat podmínku $-1 \le x \le 1$.

Ve středoškolské matematice nám postačí tato pravidla. Vyzkoušet si to můžeme na následujících příkladech.

Příklad 35

Určete definiční obor funkcí:

a) $f(x)=\frac{\ln(x+5)}{x^2-5x+6}$,

b) $g(x)=\frac{\sin(x)}{\sin(x)-\cos(x)}$,

c) $h(x)=\sqrt{\frac{x+5}{x-2}}$.

V prvním příkladě se jedná o podmínku 1) a 3). Využijeme funkce Reduce[]:

```
In[1]:= Reduce[(x + 5) > 0 && x^2 - 5 x + 6 ≠ 0, x] // TraditionalForm
Out[1]//TraditionalForm=
-5 < x < 2 ⋁ 2 < x < 3 ⋁ x > 3
```

Obrázek 75 – Příklad 35 – zadání a)

V druhém příkladě se jedná o podmínku 1). Zde je vhodné vyhledat v jakých hodnotách funkce není definovaná.

```
In[1]:= Solve[Sin[x] - Cos[x] == 0, x]

  Solve::ifun :
    Inverse functions are being used by Solve, so some solutions may not be found;
      use Reduce for complete solution information. >>

Out[1]=
  {{x → -3π/4}, {x → π/4}}

In[2]:= Reduce[Sin[x] - Cos[x] == 0, x, Reals]

Out[2]=
  C[1] ∈ Integers &&
    (x == -2 ArcTan[1 + √2] + 2 π C[1] || x == -2 ArcTan[1 - √2] + 2 π C[1])
```

Obrázek 76 – Příklad 35 – zadání b)

V tomto případě nám přehlednější výsledek dá funkce Solve[], ačkoli pouze v intervalu od $-\pi$ do π. Přesný výsledek funkce Reduce[] je málo čitelný. Podle výsledků do definičního oboru nepatří $x \neq -\frac{3\pi}{4} + 2\pi k, \frac{\pi}{4} + 2\pi k;\ k \in \mathbf{Z}$,

lépe zapsaný výsledek podle zvyklostí je: $x \neq \frac{\pi}{4} + 2\pi k, \frac{5\pi}{4} + 2\pi k;\ k \in \mathbf{Z}$.

Třetí příklad je kombinací podmínky 1) a 2). Stačí prozkoumat jen podmínku 2), která v sobě skrývá už podmínku 1).

```
In[1]:= Reduce[(x + 5)/(x - 2) ≥ 0, x] // TraditionalForm

Out[1]//TraditionalForm=
  x ≤ -5 ∨ x > 2
```

Obrázek 77 – Příklad 35 – zadání c)

Další důležitou vlastností je, zda je funkce sudá nebo lichá. Můžete vyjít z definice takové funkce. Pro sudou funkci v celém Df platí $f(-x) = f(x)$ a pro lichou $f(-x) = -f(x)$. Využijte v tomto případě funkci Symplify[], která zjednoduší zadanou rovnost. Pokud tato rovnost platí v R, vypíše funkce hodnotu True. Jinak vypíše rovnost, kterou musí proměnná splňovat.

Příklad 36

Určete, která ze zadaných funkcí je sudá, případně lichá.

a) $f(x) = \frac{x^2}{1+x^2}$,

b) $g(x) = x^3 - x$,

c) $h(x) = \frac{x+1}{1+x^2}$.

```
In[1]:= f[x_] = x^2/(1 + x^2);
        Simplify[f[-x] == f[x]]

Out[2]=
        True
```

Obrázek 78 – Příklad 36 – zadání a)

Funkce $f(x)$ je sudá.

```
In[3]:= g[x_] = x^3 - x;
        Simplify[g[-x] == -g[x]]

Out[4]=
        True
```

Obrázek 79 – Příklad 36 – zadání b)

Funkce $g(x)$ je lichá.

```
In[5]:= h[x_] = (x + 1)/(1 + x^2);
        Simplify[h[-x] == h[x]]

Out[6]=
        x/(1 + x^2) == 0

In[7]:= Simplify[h[-x] == -h[x]]

Out[7]=
        1/(1 + x^2) == 0
```

Obrázek 80 – Příklad 36 – zadání c)

Funkce $h(x)$ není sudá ani lichá, protože výsledek rovností není True. Ani jedna podmínka neplatí pro celý definiční obor, jen pro x z výsledné rovnosti.

Podobným postupem ověříme periodičnost funkce, pokud danou periodu známe. Pro periodickou funkci $f(x)$ s periodou a platí: $f(x+a) = f(x)$.

Příklad 37

Potvrďte, že funkce sin $(2x)$ má periodu π.

Prvním příkazem Simplify[] si potvrdíme, že π je jednou z period funkce. Příkaz Reduce[] nám pomůže zjistit, zda je π nejmenší periodou.

```
In[1]:= f[x_] = Sin[2 x];
        Simplify[f[x + π] == f[x]]

Out[2]=
        True
```

Obrázek 81 – Příklad 37 – ověření periody π

```
In[3]:= Reduce[f[x + a] == f[x], a]
```

Out[3]=

$$C[1] \in \text{Integers} \;\&\&\; \left(a == -x + \frac{1}{2}\,(\pi - \text{ArcSin}[\text{Sin}[2x]] + 2\pi C[1]) \;||\; a == -x + \frac{1}{2}\,(\text{ArcSin}[\text{Sin}[2x]] + 2\pi C[1])\right)$$

Obrázek 82 – Příklad 37 – hledání nejmenší periody

Dostáváme dva výsledky s celočíselným parametrem. Po úpravách dostaneme:

$a = -2x + \frac{\pi}{2} + \pi k; k \in \mathbf{Z}$,

$a = \pi k; k \in \mathbf{Z}$.

Jen druhý výsledek je nezávislý na x a splňuje podmínky periody. Tím jsme potvrdili, že π je základní perioda funkce sin (2x).

Další důležitou vlastností funkcí je monotónnost funkce. Opět využijte funkci Simplify[]. Funkce $f(x)$ je rostoucí na daném intervalu, pokud platí: $f(x_1) < f(x_2)$ za podmínky $x_1 < x_2$.

Funkce $f(x)$ je klesající na daném intervalu, pokud platí: $f(x_2) < f(x_1)$ za podmínky $x_1 < x_2$.

Příklad 38

Rozhodněte, která ze zadaných funkcí je rostoucí a která klesající:

a) $f(x) = 3x + 1$,

b) $g(x) = \sqrt{1 - 3x}$,

c) $h(x) = \frac{x-1}{x+2}$.

Pro názornost si zobrazíme i graf dané funkce.

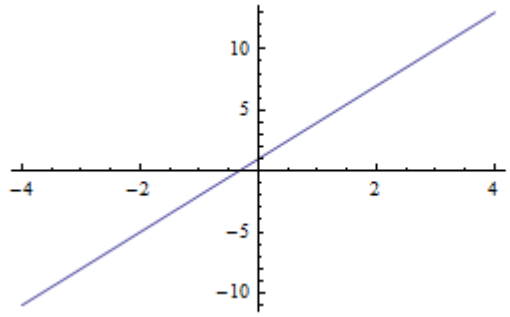
```
In[1]:= f[x_] = 3 x + 1;
        Simplify[f[x1] < f[x2], x1 < x2]
```

Out[2]=

```
True
```

```
In[3]:= Plot[f[x], {x, -4, 4}]
```

Out[3]=

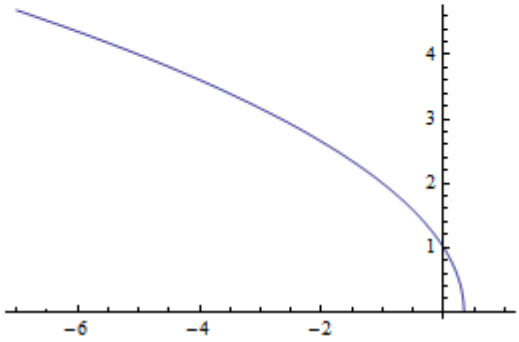


Obrázek 83 – Příklad 38 – zadání a)

Výsledek True je potvrzením předpokladu, že funkce je rostoucí. To potvrzuje i graf funkce.

```
In[4]:= g[x_] = √(1 - 3 x);
        Simplify[g[x2] < g[x1], x1 < x2]

Out[5]=
        √(1 - 3 x2) < √(1 - 3 x1)

In[6]:= Plot[g[x], {x, -7, 1}]
```

Out[6]=



Obrázek 84 – Příklad 38 – zadání b), první zobrazení

Pokud omezíme zadání na definiční obor $x \leq \frac{1}{3}$; potvrdí se nám klesající charakter funkce na celém definičním oboru.

```
In[7]:= Simplify[g[x2] < g[x1], {x1 < x2, x2 ≤ 1/3}]

Out[7]=
        True
```

Obrázek 85 – Příklad 38 – zadání b), omezení na definiční obor

Výsledek True znamená, že na celém definičním oboru funkce klesá.

Funkce $h(x)$ není definována v bodě -2. Proto ověření rozdělme na dva úseky.

```
In[8]:= h[x_] = (x - 1)/(x + 2);
        Simplify[h[x1] < h[x2], {x1 < x2, x2 < -2}]

Out[9]=
        True
```

Obrázek 86 – Příklad 38 – zadání c), první interval

Na intervalu $(\infty,-2)$ je funkce rostoucí.

```
In[10]:=
        Simplify[h[x1] < h[x2], {x1 < x2, x1 > -2}]

Out[10]=
        True
```

Obrázek 87 – Příklad 38 – zadání c), druhý interval

Na intervalu $(-2, \infty)$ je funkce také rostoucí.

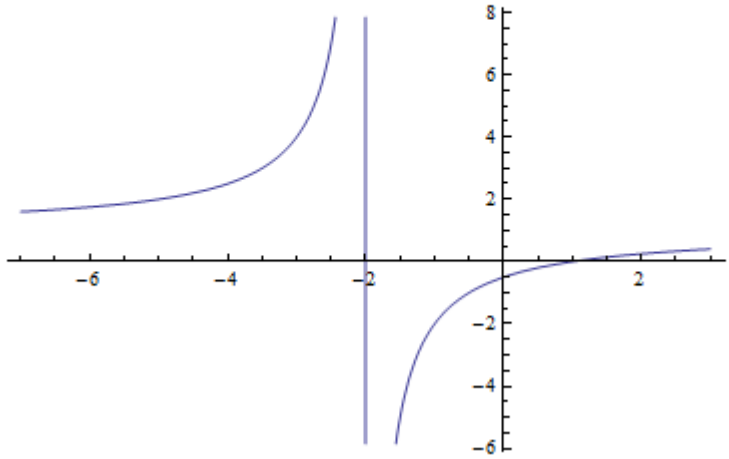
Co se stane, když první číslo vezmeme z prvního intervalu a druhé z druhého.

```
In[11]:=
Simplify[h[x1] < h[x2], {x1 < x2, x1 < -2, x2 > -2}]
Out[11]=
False
```

Obrázek 88 – Příklad 38 – zadání c), oba intervaly

Samostatně je funkce na jednotlivých intervalech rostoucí, ale v celém definičním oboru to neplatí. Ukážeme si to na grafu.

```
In[12]:=
Plot[h[x], {x, -7, 3}]
Out[12]=
```



Obrázek 89 – Příklad 38 – zadání c), graf

Z grafu jde vidět, že je funkce prostá. To znamená, že pro různá x nabývá různé hodnoty. Neexistuje tedy dvojice různých čísel $x1$ a $x2$, jejichž funkční hodnoty $h(x1), h(x2)$ jsou shodné. Důkaz, že je funkce prostá, nám dá také funkce Simplify[].

```
In[13]:=
Simplify[h[x1] == h[x2], {x1 ≠ x2}]
Out[13]=
False
```

Obrázek 90 – Příklad 38 – zadání c), ověření prosté funkce

Výsledek False nám potvrzuje, že neexistují dvě různé proměnné $x1$ a $x2$, pro které by funkce $h(x)$ nabývala shodných hodnot. Tím jsme potvrdili, že funkce je prostá.

4.1 Lineární funkce

Lineární funkcí rozumíme funkci s definičním oborem **R** a zadanou předpisem:

$f : y = ax + b$,

kde a a b jsou reálná čísla. Grafem takové funkce je přímka.

Vytvořit graf takové funkce příkazem Plot[] už umíme. V následujícím příkladu si ukážeme několik základních postupů při hledání lineární funkce daných vlastností.

Příklad 39

Nalezněte lineární funkci, která prochází body $[2,3]$, $[5,-3]$.

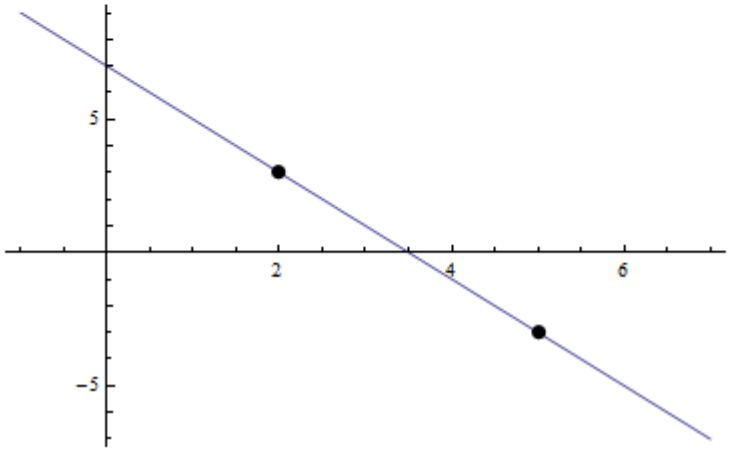
Nadefinujeme si lineární funkci $f(x)$ s parametry *a* a *b*. Nyní můžeme vyřešit soustavu rovnic $f(2)=3$, $f(5)=-3$ s neznámými parametry funkce.

```
In[1]:= f[x_] = a * x + b;
        Solve[{f[2] == 3, f[5] == -3}, {a, b}]

Out[2]=
  {{a → -2, b → 7}}
```

Obrázek 91 – Příklad 39 – výpočet parametrů

Dosadíme parametry do funkce jejím předefinováním v buňce In[3] a nakonec zobrazíme graf s vyznačenými body.

```
In[1]:= f[x_] = a * x + b;
        Solve[{f[2] == 3, f[5] == -3}, {a, b}]

Out[2]=
  {{a → -2, b → 7}}

In[3]:= f[x_] = f[x] /. {a → -2, b → 7}

Out[3]=
  7 - 2 x

In[4]:= Plot[f[x], {x, -1, 7},
         Epilog → {PointSize[0.02], Point[{2, 3}], Point[{5, -3}]}]

Out[4]=
```



Obrázek 92 – Příklad 39 – dosazení parametrů a zobrazení grafu

Parametry *a* a *b* jsme mohli najít i jako řešení soustavy rovnic $2a+b=3;5a+b=-3$ a funkci definovat až po tomto výpočtu.

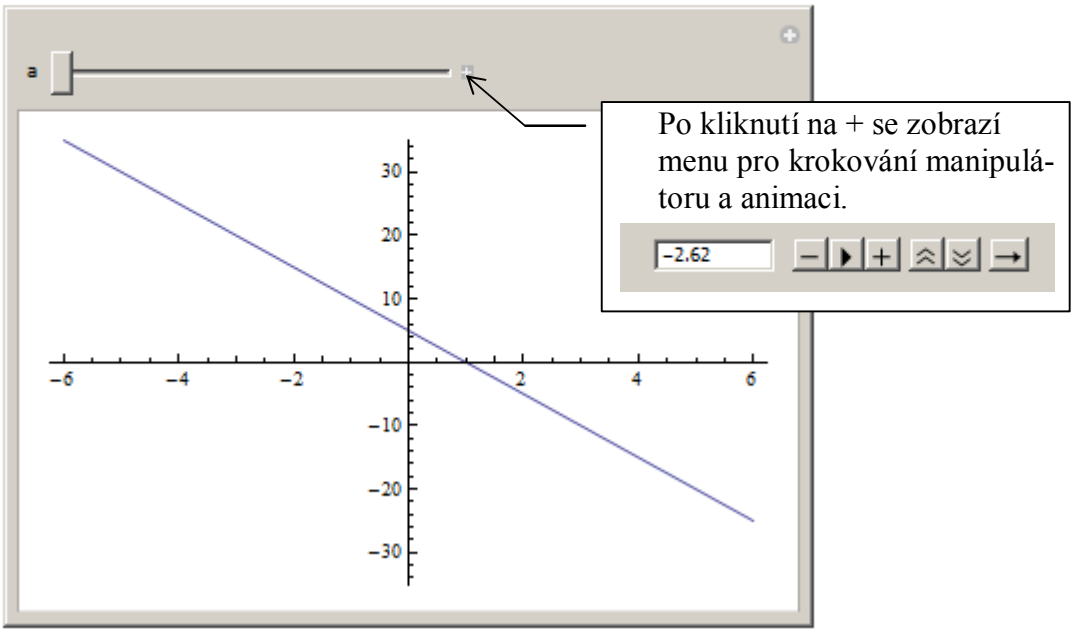
Příkazy se hlavní myšlenkou neliší, záleží proto jen na vás, který zápis postupu se vám líbí víc.

```
In[1]:= Solve[{2 a + b == 3, 5 a + b == -3}, {a, b}]
Out[1]=
  {{a → -2, b → 7}}

In[2]:= f[x_] = -2 x + 7
Out[2]=
  7 - 2 x
```

Obrázek 93 – Příklad 39 – druhý postup

Na dalším příkladu uvidíme užití funkce Manipulate[]. Ta umožňuje do funkcí přidat manipulátory pro volnější zadávání parametrů. Pojďme si ukázat, jak zobrazit grafy lineárních funkcí s parametrem *b* = 5 a parametrem *a* v rozsahu od -5 do 5.

```
In[1]:= Manipulate[Plot[a * x + 5, {x, -6, 6}, PlotRange → {-35, 35}],
         {a, -5, 5}]
Out[1]=
```

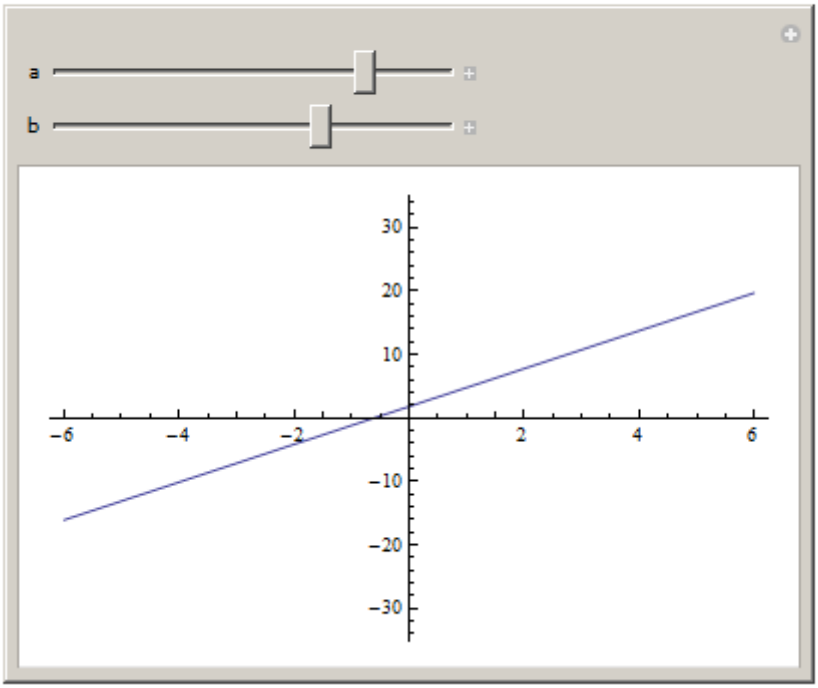


Obrázek 94 – manipulátor

Takto vytvořený model umožňuje animaci všech vzniklých funkcí. Pevné nastavení rozsahu osy *y* je pro lepší efekt animace při manipulaci s hodnotou parametru. Osy se v takovém případě nemění. Velmi jednoduše přidáme manipulátor i pro parametr *b*. Na tomto příkladě si můžeme ukázat, jak se mění sklona posunutí funkce v závislosti na hodnotách parametru *a* a *b*.

```
In[2]:= Manipulate[Plot[a * x + b, {x, -6, 6}, PlotRange → {-35, 35}],
        {a, -5, 5}, {b, -5, 5}]
```

Out[2]=

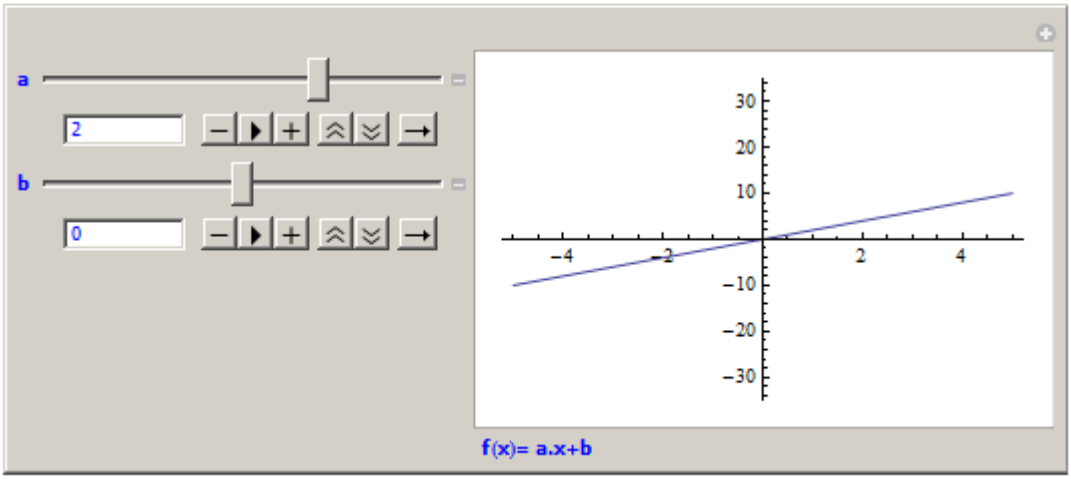


Obrázek 95 – manipulátory pro oba parametry

Drobným vylepšením je popisek manipulátoru direktivou FrameLabel. Styl popisku určíme direktivou LabelStyle. Výchozí nastavení manipulátoru uvedeme jako seznam se jménem proměnné { *a*,1 ,–5, 5 , tímto příkazem nadefinujeme počáteční hodnotu parametru *a* na 1 v rozsahu od -5 do 5. Zobrazení posuvníků manipulátorů stačí určit zobrazovací direktivou Appearance. Umístění manipulátorů vlevo nastavíme direktivo ControlPlacement.

```
In[1]:= Manipulate[Plot[a * x + b, {x, -5, 5}, PlotRange → {-35, 35}],
        {{a, 2}, -5, 5, Appearance → "Open"},
        {{b, 0}, -5, 5, Appearance → "Open"}, FrameLabel → "f(x)= a.x+b",
        LabelStyle → Directive[Blue, Bold], ControlPlacement → Left]
```

Out[1]=



Obrázek 96 – nastavení manipulátoru

Příklad 40

Graficky zobrazte řešení soustavy lineárních rovnic $2y-3x-5=0$, $y+x-5=0$.

Prvním krokem je vytvoření lineárních funkcí z jednotlivých rovnic:

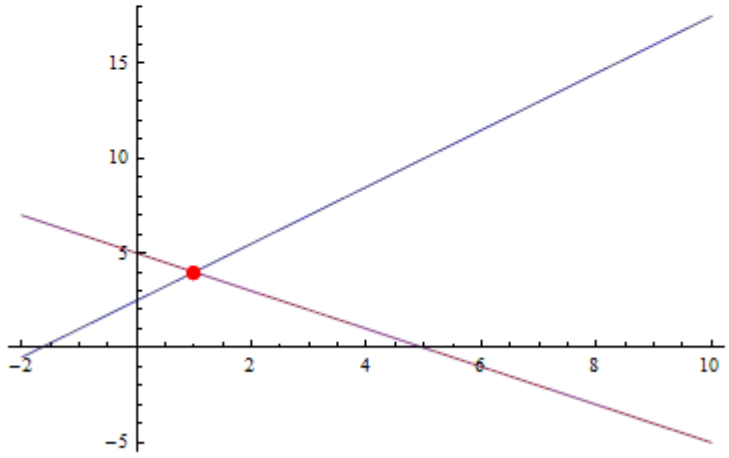
$f1:\ y=\frac{3}{2}x+\frac{5}{2}$,

$f2:\ y=-x+5$.

Zobrazit grafy obou funkcí není s Plot[] žádný problém. Nám se hodí vypočítat souřadnice průsečíku funkci pomocí Solve[], výsledek si uložíme do proměnné *reseni*. Direktivou /. tyto hodnoty dosadíme do uspořádané dvojice $\{x,y\}$ přestavující průsečík grafů funkcí, tuto dvojici označíme proměnnou *bod*. Grafy funkcí označené *graf1* a graf s průsečíkem *graf2*. Spojíme je do jednoho obrázku příkazem Show[].

```
In[1]:= f1[x_] = 3/2 x + 5/2;
        f2[x_] = -x + 5;
        reseni = Solve[{f1[x] == f2[x], y == f1[x]}, {x, y}];
        bod = {x, y} /. reseni;
        graf1 = Plot[{f1[x], f2[x]}, {x, -2, 10}];
        graf2 = ListPlot[bod, PlotStyle → {Red, PointSize[0.02]}];
        Show[graf1, graf2]
        reseni
```

Out[7]=



Out[8]=

```
{{x → 1, y → 4}}
```

Obrázek 97 – Příklad 40 – zobrazení grafu a výsledku

4.2 Kvadratické funkce

Kvadratickou funkcí rozumíme funkci s definičním oborem **R** a zadanou předpisem:

$f: y=a\cdot x^2+b\cdot x+c$,

kde *a*, *b, c* jsou reálná čísla. Grafem takové funkce je parabola.

Příklad 41

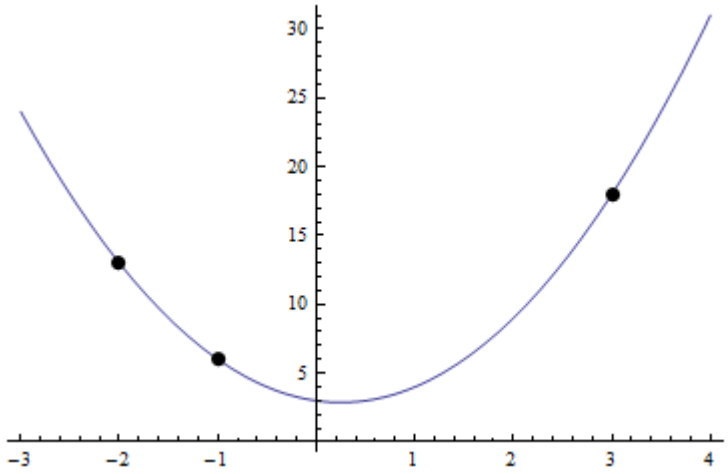
Určete kvadratickou funkci procházející body $[3,18]$, $[-1,6]$, $[-2,13]$.

Pomocí funkce Solve[] vypočítáme hodnoty parametrů *a*, *b*, *c*. Výsledek dosadíme pomocí direktivy */.* do předefinované funkce *f*(*x*). Výslednou funkci zobrazíme grafem s vyznačenými body.

```
In[1]:= f[x_] = a x^2 + b x + c;
        vysledek = Solve[{f[3] == 18, f[-1] == 6, f[-2] == 13},
           {a, b, c}];
        f[x_] = f[x] /. vysledek
        Plot[f[x], {x, -3, 4},
         Epilog → {PointSize[0.02], Point[{3, 18}], Point[{-1, 6}],
           Point[{-2, 13}]}]
```

```
Out[3]=
{3 - x + 2 x^2}
```

Out[4]=



Obrázek 98 – Příklad 41

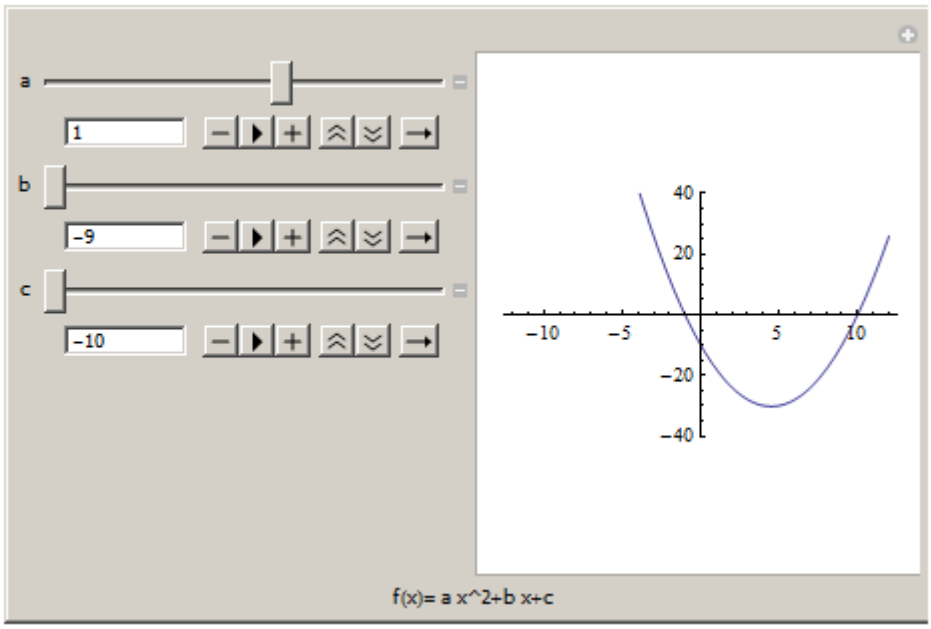
Jaký vliv může mít hodnota parametru a, b, c na graf kvadratické funkce, si vyzkoušíme formou aplikace s manipulátory.

```
In[1]:= Manipulate[Plot[a * x^2 + b * x + c, {x, -12, 12},
          PlotRange → {-40, 40}], {{a, 1}, -5, 5, Appearance → "Open"},
        {b, -9, 9, Appearance → "Open"},
        {c, -10, 10, Appearance → "Open"},
        FrameLabel → "f(x)= a x^2+b x+c", ControlPlacement → Left]
```

Obrázek 99 – příkaz pro manipulátor kvadratické funkce

Výsledek příkazu vidíte na následujícím obrázku. Můžeme si vyzkoušet, jak se změní tvar funkce při změně znaménka parametru *a*. Zajímavý je i posun při změně parametru *b*.

Out[1]=



Obrázek 100 – výsledek příkazu s manipulátory pro kvadratickou funkci.

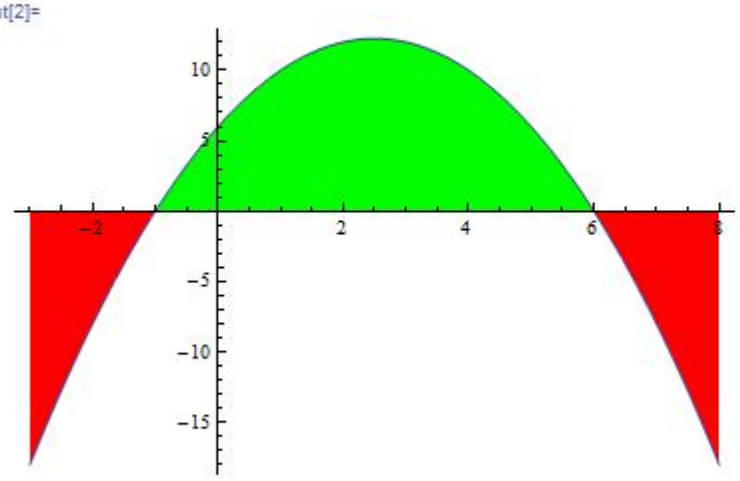
Grafy kvadratické funkce nám mohou pomoci při řešení kvadratických nerovnic. Můžeme se pokusit zobrazit, kdy funkce nabývá kladných a kdy záporných hodnot.

Příklad 42

Vyřešte pomocí grafu kvadratické funkce nerovnost $-x^2+5x+6>0$.

```
In[1]:= f[x_] = -x^2 + 5 x + 6;
        Plot[f[x], {x, -3, 8}, Filling → {1 → {Axis, {Red, Green}}}]
        Reduce[f[x] > 0, x] // TraditionalForm
```

Out[2]=



Out[3]//TraditionalForm=

$$-1 < x < 6$$

Obrázek 101 – Příklad 42

Barvy bohužel nejsou určeny znaménkem funkční hodnoty, ale pravidelně se střídají podle přechodu přes hraniční hodnotu, v tomto případě přes osu. Grafy funkcí jsou základem i ná-

sledující úlohy, kdy se pokusíme ukázat řešení soustavy rovnic pomocí grafu kvadratické a lineární funkce.

Příklad 43

Zobrazte graficky řešení soustavy rovnic $x-y-1=0$, $x^2+2x-y-13=0$.

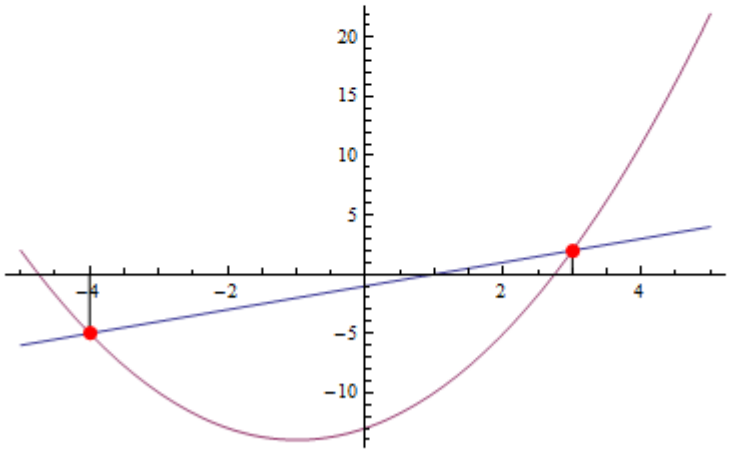
Nejprve si vytvoříme funkce vyjadřující z rovnic proměnnou *y*:

$f1: y=x-1,$ $f2: y=x^2+2x-13.$

Pro vyznačení řešení budeme muset vypočítat i průsečíky funkcí pomocí Solve[]. Opět sestavíme z řešení direktivou /. sadu bodů s označením *body*. Zobrazíme dva grafy (*graf1*, *graf2*), které spojíme příkazem Show[].

```
In[1]:= f1[x_] = x - 1;
        f2[x_] = x^2 + 2 x - 13;
        reseni = Solve[f1[x] == f2[x], x];
        body = {x, f1[x]} /. reseni;
        graf1 = Plot[{f1[x], f2[x]}, {x, -5, 5}];
        graf2 = ListPlot[body, PlotStyle → {Red, PointSize[0.02]},
           Filling → {1 → {Axis, {Black}}}];
        Show[graf1, graf2]
        body
```

Out[7]=




```
Out[8]=
{{-4, -5}, {3, 2}}
```

Obrázek 102 – Příklad 43

Zajímavé je i použití programu MATHEMATICA pro řešení kvadratických rovnic s parametrem.

Příklad 44

Určete řešení rovnice $(1-k^2)x^2-2kx-1=0$ v závislosti na parametru $k \in \mathbf{R}$.

```
In[1]:= Reduce[(1 - k^2) x^2 + 2 k * x + 1 == 0, {k, x}] // TraditionalForm
```

Out[1]//TraditionalForm=

$$\left((k=-1 \vee k=1) \wedge x=-\frac{k}{2}\right) \vee \left(k^2-1 \neq 0 \wedge \left(x=\frac{k-\sqrt{2k^2-1}}{k^2-1} \vee x=\frac{\sqrt{2k^2-1}+k}{k^2-1}\right)\right)$$

Obrázek 103 – Příklad 44

Výsledek si zaslouží interpretaci. Pro $k=\pm 1$ dostaneme lineární rovnici, pro ostatní možné hodnoty k dostáváme kvadratickou rovnici. Bohužel z výsledku nevidíme, pro která k rovnice nemá řešení, má pouze jedno řešení, nebo má dvě různá řešení. Tento výsledek nám ale pomůže. Můžete k soustavě nerovností v příkazu Reduce[] přidat podmínku pro hodnotu diskriminantu. To je ta část pod odmocninou. Protože jednotlivé části jsou moc rozsáhlé. Uvedeme zde nejprve tu část pro hodnotu diskriminantu 0, jedno dvojnásobné řešení.

```
In[2]:= Reduce[{(1 - k^2) x^2 + 2 k * x + 1 == 0, k^2 - 1 ≠ 0, k/(k^2 - 1) == x}, {k, x}] //
        TraditionalForm
```

Out[2]//TraditionalForm=

$$\left(k=-\frac{1}{\sqrt{2}} \vee k=\frac{1}{\sqrt{2}}\right) \wedge x=-2k$$

Obrázek 104 – Příklad 44 – pro D = 0

Další část redukujeme jen na hledání hodnoty parametru, pro který rovnice nemá reálné řešení.

```
In[3]:= Reduce[{k^2 - 1 ≠ 0, 2 k^2 - 1 < 0}, k] // TraditionalForm
```

Out[3]//TraditionalForm=

$$-\frac{1}{\sqrt{2}}<k<\frac{1}{\sqrt{2}}$$

Obrázek 105 – Příklad 44 – pro D < 0

Zbylo hledání hodnoty parametru k pro dva různé kořeny rovnice.

```
In[4]:= Reduce[{k^2 - 1 ≠ 0, 2 k^2 - 1 > 0}, k, Reals] // TraditionalForm
```

Out[4]//TraditionalForm=

$$k<-1 \vee -1<k<-\frac{1}{\sqrt{2}} \vee \frac{1}{\sqrt{2}}<k<1 \vee k>1$$

Obrázek 106 – Příklad 44 – pro D > 0

Pokud bychom chtěli vypsat i výrazy pro výpočet x, byl by výpis moc dlouhý a nepřehledný. Jednoduše do předchozích ukázek přidáme původní rovnici, do seznamu proměnných doplníme x a zrušíme omezení jen na reálná čísla.

4.3 Mocninné funkce

V této podkapitole se podíváme na základní mocninné funkce typu

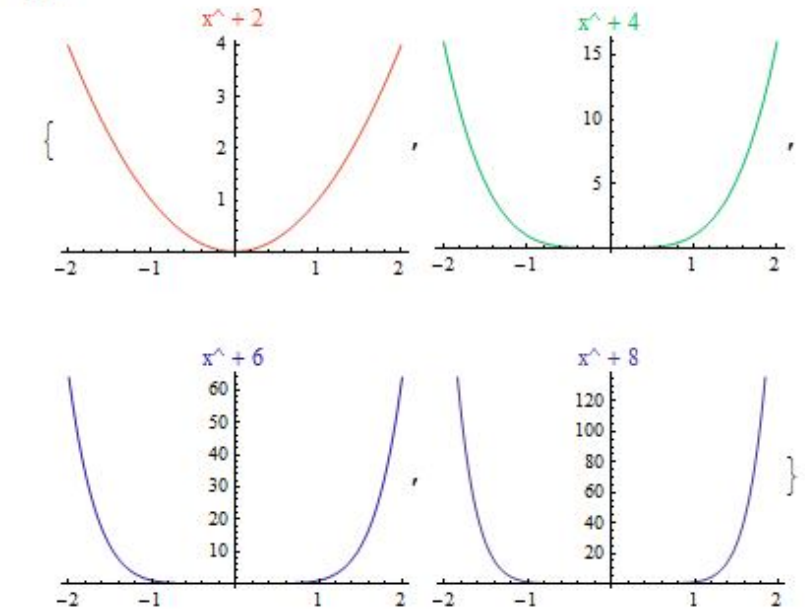
$f(x)=x^n,\ n\in\mathbf{N}$ a $f(x)=x^{\frac{1}{n}},\ n\in\mathbf{N}$.

Příklad 45

Zobrazte grafy funkcí $f(x)=x^{2n},\ n\in\{1,2,3,4\}$. Nejprve samostatně, pak společně.

```
In[1]:=
grafy =
 Table[Plot[x^(2 n), {x, -2, 2},
   PlotStyle → RGBColor[r = Random[], g = Random[], b = Random[]],
   PlotLabel → Style["x^" <> 2 n, RGBColor[r, g, b]]], {n, 1, 4}]
```

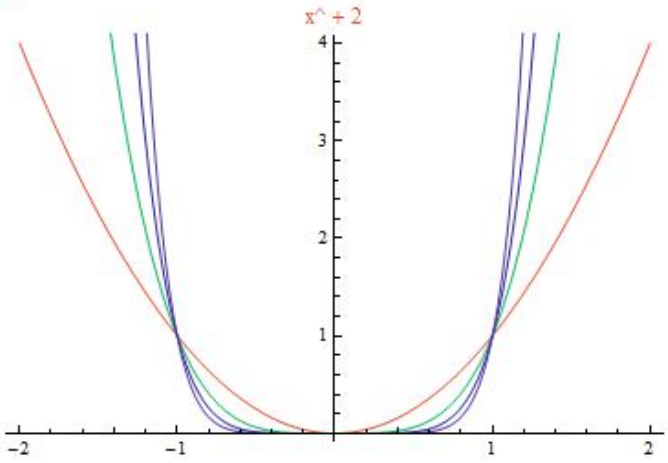
Out[1]=



Obrázek 107 – Příklad 45 – samostatné grafy

```
In[2]:= Show[grafy]
```

Out[2]=



Obrázek 108 – Příklad 45 – společný graf

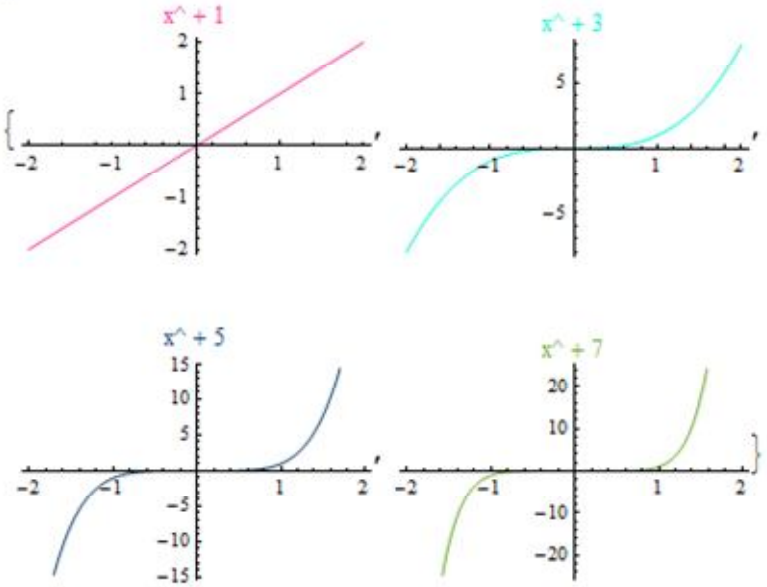
Program zatím neumí popsat jednotlivé křivky. Graf lze opatřit popiskem zobrazovací direktivou PlotLabel, při sloučení se ale popisky překryjí. Z grafů můžeme vypozorovat, že grafy funkcí se sudou mocninou přecházejí z paraboly do tvaru písmene U.

Příklad 46

Zobrazte grafy funkcí $f(x)=x^{2n-1}$, $n \in \{1,2,3,4\}$. Nejprve samostatně, pak společně.

```
In[1]:=
grafy =
 Table[Plot[x^(2 n - 1), {x, -2, 2},
   PlotStyle → RGBColor[r = Random[], g = Random[], b = Random[]],
   PlotLabel → Style["x^" <> 2 n - 1, RGBColor[r, g, b]]], {n, 1, 4}]
```

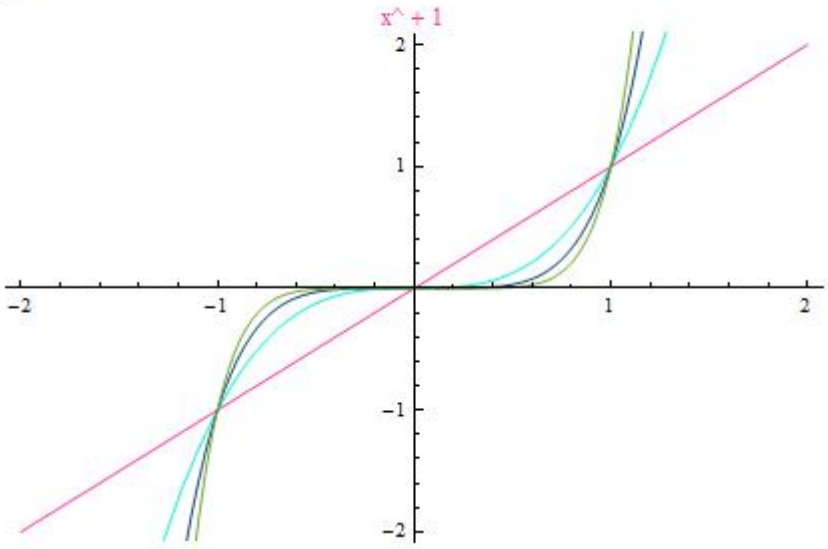
Out[1]=



Obrázek 109 – Příklad 46 – samostatné grafy

```
In[2]:= Show[grafy]
```

Out[2]=



Obrázek 110 – Příklad 46 – společný graf

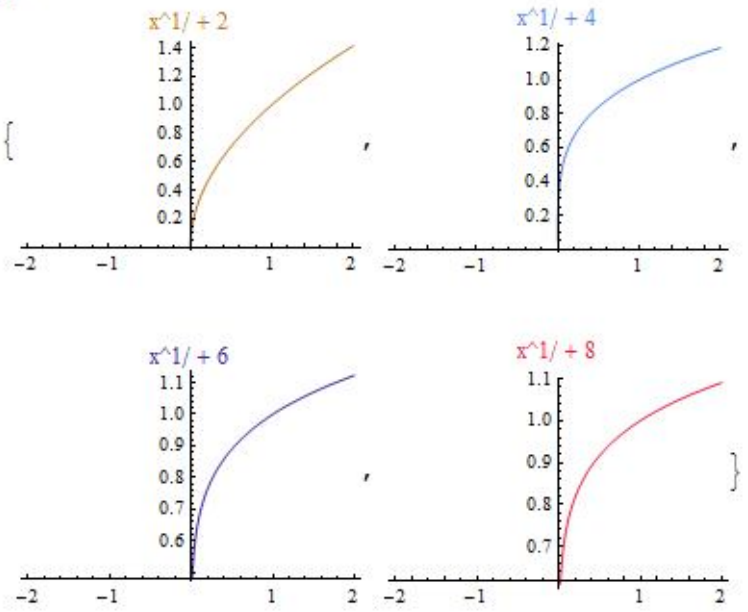
Problematičtější situace je u odmocnin. Program MATHEMATICA neumí vypočítat lichou odmocninu ze záporného čísla.

Příklad 47

Zobrazte grafy funkcí $f(x)=\sqrt[2n]{x}$, $n \in \{1,2,3,4\}$.

```
In[1]:=
grafy =
 Table[Plot[Sqrt[x, 2 n], {x, -2, 2},
   PlotStyle → RGBColor[r = Random[], g = Random[], b = Random[]],
   PlotLabel → Style["x^1/" + 2 n, RGBColor[r, g, b]]], {n, 1, 4}]
```

Out[1]=



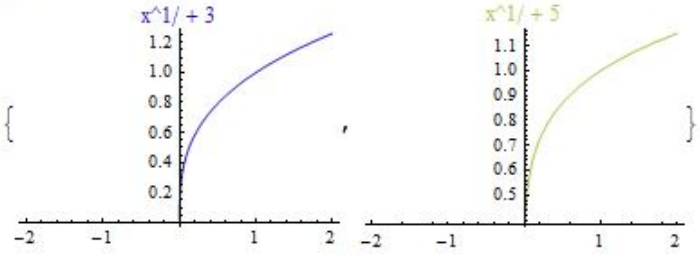
Obrázek 111 – Příklad 47

Příklad 48

Zobrazte grafy funkcí $f(x)=\sqrt[2n+1]{x}$, $n \in \{1,2\}$.

```
In[1]:=
grafy =
 Table[Plot[Sqrt[x, 2 n + 1], {x, -2, 2},
   PlotStyle → RGBColor[r = Random[], g = Random[], b = Random[]],
   PlotLabel → Style["x^1/" + 2 n + 1, RGBColor[r, g, b]]], {n, 1, 2}]
```

Out[1]=

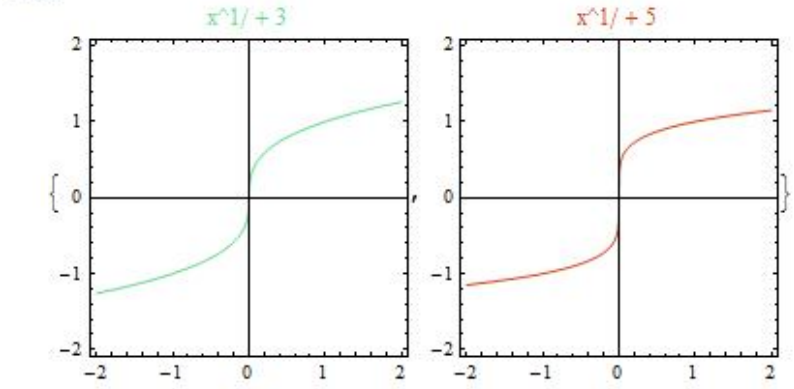


Obrázek 112 – Příklad 48

V tomto příkladu chybí výsledek pro záporná čísla. V takovém případě použijeme graf pro implicitně zadanou funkci ContourPlot[]. Taková funkce $y = \sqrt[2n+1]{x}$ se dá zapsat v implicitní podobě jako $x = y^{2n+1}$.

```
In[1]:=
grafy =
 Table[ContourPlot[x == y^(2 n + 1), {x, -2, 2}, {y, -2, 2},
   ContourStyle → RGBColor[r = Random[], g = Random[], b = Random[]],
   PlotLabel → Style["x^1/" <> 2 n + 1, RGBColor[r, g, b]], Axes → True],
  {n, 1, 2}]
```

Out[1]=



Obrázek 113 – Příklad 48 – implicitní zadání

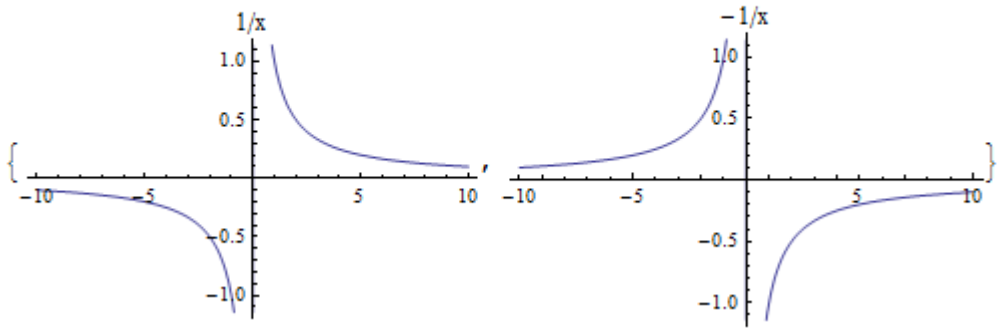
Pozor, na problém vypočítání liché odmocniny můžeme v programu MATHEMATICA narazit na více místech, speciálně u zobrazení grafů!

4.4 Lomené funkce

Základní lomené funkce jsou $f1(x) = \frac{1}{x}$ a $f2(x) = -\frac{1}{x}$.

```
In[1]:= Table[Plot[n / x, {x, -10, 10}, PlotLabel → "1/x" * n], {n, 1, -1, -2}]
```

Out[1]=



Obrázek 114 – základní lomené funkce

Posunutím ve směru jednotlivých os můžeme u obecných lomených funkcí zjistit úpravou zápisu funkce metodu Apart[]. Ta provede vydělení výrazů a výsledek zapíše ve formě co nejjednodušších jmenovatelů. Pro ukázku použijeme funkci: $f(x) = \frac{2x+3}{1+x}$.

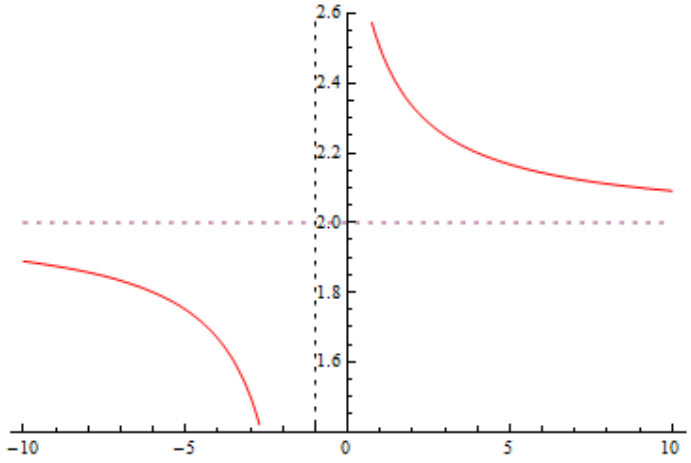
```
In[1]:= f[x_] = (2 x + 3)/(x + 1)

Out[1]=
(3 + 2 x)/(1 + x)

In[2]:= Apart[f[x]]

Out[2]=
2 + 1/(1 + x)
```

Obrázek 115 – zjištění posunutí lomené funkce

V této ukázce je daná funkce ve směru osy *x* posunutá o -1 a ve směru osy *y* o +2. Díky této informaci doplníme do grafu asymptoty. Svislou asymptotu nastavíme direktivou Exclusions a ExclusionsStyle.

```
In[3]:= Plot[{f[x], 2}, {x, -10, 10}, Exclusions → {x == -1},
        ExclusionsStyle → Dashing[Small], PlotStyle → {Red, Dashing[Small]}]
```

Out[3]=



Obrázek 116 – zobrazení asymptot

4.5 Exponenciální a logaritmické funkce a výrazy

Zobrazení grafu exponenciálních funkcí je jednoduché. Opět použijeme manipulátorů, aby bylo vidět, jak se funkce mění v závislosti na jednotlivých parametrech.

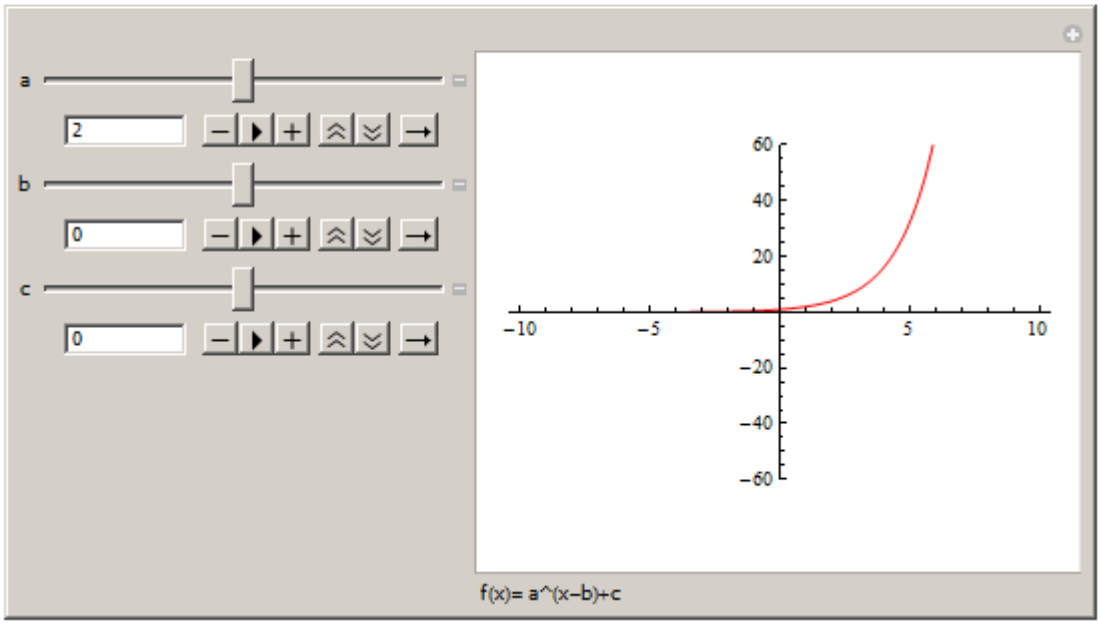
Základní exponenciální funkce má tvar $f(x): y = a^x$, kde $x \in \mathbf{R}$, $a > 0$. My se pokusíme zobrazit pomocí manipulátorů funkci s posuny v obou osách. Naše funkce bude mít tvar:

$f(x): y = a^{x-b} + c$ kde $x \in \mathbf{R}$, $a > 0$.

Parametr *b* představuje posun v ose *x* a parametr *c* představuje posun v ose *y*. Znázorňujeme v tomto případě dvě funkce. Druhá funkce je vlastně asymptotou ve tvaru $y = c$.

```
In[1]:= Manipulate[Plot[{a^(x-b) + c, c}, {x, -10, 10}, PlotStyle → {Red, Dashing[Small]},
  PlotRange → {-60, 60}], {{a, 2}, 0, 4, Appearance → "Open"},
 {{b, 0}, -5, 5, Appearance → "Open"}, {{c, 0}, -20, 20, Appearance → "Open"},
 FrameLabel → "f(x)= a^(x-b)+c", ControlPlacement → Left]
```

Out[1]=



Obrázek 117 – exponenciální funkce

Na následujících příkladech si ukážeme, že pro výpočet exponenciální rovnic je vhodnější metoda Reduce[] než Solve[], ale v obou případech dojdeme ke správnému řešení. Zároveň uvidíme, že program MATHEMATICA díky symbolickým výpočtům dokáže výsledky zapsat i netypických zápisech.

Příklad 49

Řešte v **R** rovnice: a) $\left(\frac{25}{9}\right)^x = \left(\frac{3}{5}\right)^3$, b) $2^x \cdot 5^{x+1} = 50$.

```
In[1]:= Solve[(25/9)^x == (3/5)^3, x]

Solve::ifun : Inverse functions are being used by Solve, so
    some solutions may not be found; use Reduce for complete solution information. >>

Out[1]= {{x → -3/2}}

In[2]:= Reduce[(25/9)^x == (3/5)^3, x, Reals]

Out[2]= x == -3/2
```

Obrázek 118 – Příklad 49 – zadání a)

Program nás upozornil, že metodou Solve[] nedostaneme všechny výsledky a nabízí metodu Reduce[]. Tu je ale vhodné omezit jen na obor **R**, protože pracuje v širším oboru. Pamatuj-

me si, že výsledek Solve[] lze dál dosazovat do výrazů a výsledek Reduce[] je vlastně rovnicí, kterou také můžeme následně využít.

```
In[1]:= Solve[2^x * 5^(x+1) == 50, x]

  Solve::ifun : Inverse functions are being used by Solve, so
      some solutions may not be found; use Reduce for complete solution information. >>

Out[1]=
  {{x → (-Log[5] + Log[50]) / (Log[2] + Log[5])}}

In[2]:= Solve[2^x * 5^(x+1) == 50, x] // Simplify

  Solve::ifun : Inverse functions are being used by Solve, so
      some solutions may not be found; use Reduce for complete solution information. >>

Out[2]=
  {{x → 1}}

In[3]:= Reduce[2^x * 5^(x+1) == 50, x, Reals]

Out[3]=
  x == 1
```

Obrázek 119 – Příklad 49 – zadání b)

První výsledek je zapsán složitě, ale my známe postupy na součet logaritmů. Vypadá to, jako by program výsledek nechal pře poslední úpravou, k dokončení úprav nám v tomto případě pomůže funkce Simplify[]. Přehledný výsledek dostaneme i pomocí funkce Reduce[] s omezením na reálná řešení.

Další skupinou jsou logaritmické funkce, výrazy a rovnice. Za připomenutí stojí, že přirozený logaritmus se v MATHEMATICE označuje Log[], ostatní logaritmy využívají stejnou funkci se dvěma parametry (základ a hodnota).

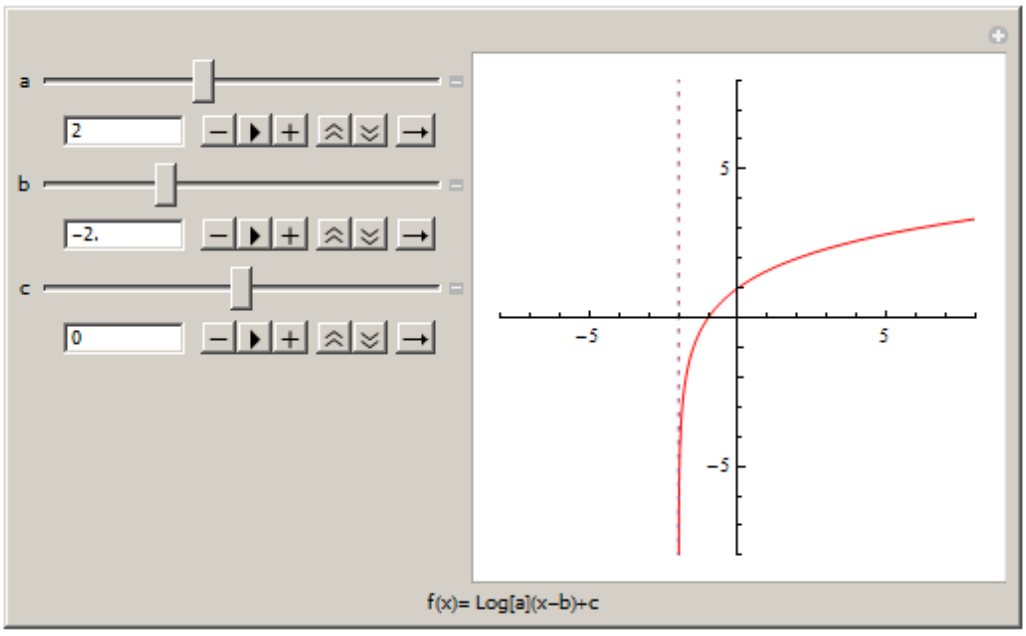
Základní logaritmická funkce má tvar $f(x): y = \log_a x$ kde $x \in \mathbf{R}^+,\ a > 0$.

My si zobrazíme manipulátorem funkci s posuny v obou osách. Naše funkce bude mít tvar: $f(x): y = \log_a(x-b)+c$, kde $x \in \mathbf{R}^+,\ a > 0$. Parametr c představuje posunutí v ose y, parametr b je posunutí v ose x a poslední parametr a představuje základ logaritmu.

Protože v případě manipulátorů nelze pro vytvoření asymptoty použít direktivu Exclusions, je vhodnější definovat parametrické zadání grafu ParametricPlot[]. První funkcí je parametrické zadání logaritmické funkce, kde souřadnice x je parametrem, druhá funkce představuje svislou asymptotu.

```
In[1]:= Manipulate[ParametricPlot[{{t, Log[a, t - b] + c}, {b, 2 t}},
          {t, -8, 8}, PlotRange → {-8, 8}, PlotStyle → {Red, Dashing[Small]}],
        {{a, 2}, 0, 5, Appearance → "Open"}, {{b, 0}, -5, 5, Appearance → "Open"},
        {{c, 0}, -20, 20, Appearance → "Open"},
        FrameLabel → "f(x)= Log[a](x-b)+c", ControlPlacement → Left]
```

Out[1]=



Obrázek 120 – logaritmická funkce s manipulátorem

Příklad 50

Vypočítejte: a) $\log_2 \log_2 16$, b) $\ln 2 + \ln 6 - \ln 12$, c) $2\log 4 + \log 75 - \log 12$.

Výrazy zapíšeme a necháme vypočítat. V případě složitějšího zápisu použijeme metodu na zjednodušení výrazu Siplify[] nebo numerický výpočet funkcí N[].

```
In[1]:= Log[2, Log[2, 16]]
Out[1]=
2
```

Obrázek 121 – Příklad 50 – zadání a)

```
In[2]:= Log[2] + Log[6] - Log[12]
Out[2]=
Log[2] + Log[6] - Log[12]
In[3]:= Log[2] + Log[6] - Log[12] // Simplify
Out[3]=
0
```

Obrázek 122 – Příklad 50 – zadání b)

V tomto případě vidíme důsledek symbolických výpočtů systému. Funkce Simplify[] postačuje k zjednodušení na výsledný výraz.

```
In[4]:= 2 Log[10, 4] + Log[10, 75] - Log[10, 12]

Out[4]=
  2 Log[4]   Log[12]   Log[75]
  -------- - -------- + --------
  Log[10]    Log[10]    Log[10]
```

```
In[5]:= 2 Log[10, 4] + Log[10, 75] - Log[10, 12] // Simplify

Out[5]=
  Log[100]
  --------
  Log[10]
```

```
In[6]:= 2 Log[10, 4] + Log[10, 75] - Log[10, 12] // N

Out[6]=
  2.
```

Obrázek 123 – Příklad 50 – zadání c)

Výsledkem je výraz $\frac{\ln 100}{\ln 10} = \log_{10} 100 = \log 100 = 2$. Pokud se naučíme správně číst výsledky funkce Simplify[], nebude nutné použít funkce N[] pro numerické vyjádření.

Příklad 51

Vypočítejte řešení v **R** rovnic:

a) $3\log 2x^2 + 2\log 3x^3 = 5\log x + 2\log 6x^3$,

b) $\log_2^2 x + 2\log_2 x - 3 = 0$,

c) $x^{\log x} = 100x$.

```
In[1]:= Solve[3 Log[10, 2 x^2] + 2 Log[10, 3 x^3] == 5 Log[10, x] + 2 Log[10, 6 x^3], x]

Out[1]=
  {{x → 1/2}}
```

Obrázek 124 – Příklad 51 – zadání a)

```
In[2]:= Solve[(Log[2, x])^2 + 2 Log[2, x] - 3 == 0, x]

Out[2]=
  {{x → 1/8}, {x → 2}}
```

Obrázek 125 – Příklad 51 – zadání b)

```
In[3]:= Solve[x^Log[10,x] == 100 x, x, Reals]

Out[3]=
  {{x → 1/10}, {x → 100}}
```

Obrázek 126 – Příklad 51 – zadání c)

Další skupinou v této kapitole jsou exponenciální a logaritmické nerovnice. Pro jejich řešení použijeme funkci Reduce[], případně N[] a Simplify[].

Příklad 52

Nalezněte řešení nerovnic v **R**:

a) $3^{x+5} > 9$, b) $3^{2x} \le 7$ c) $\log_3(x+4) \le 4$.

```
In[1]:= Reduce[3^(x+5) > 9, x, Reals]

Out[1]=
x > -3
```

Obrázek 127 – Příklad 52 – zadání a)

```
In[2]:= Reduce[3^(2 x) ≤ 7, x, Reals]

Out[2]=
x ≤ Log[7] / (2 Log[3])

In[3]:= Reduce[3^(2 x) ≤ 7, x, Reals] // N

Out[3]=
x ≤ 0.885622
```

Obrázek 128 – Příklad 52 – zadání b)

```
In[4]:= Reduce[Log[3, x + 4] ≤ 4, x, Reals]

Out[4]=
-4 < x ≤ 77
```

Obrázek 129 – Příklad 52 – zadání c)

4.6 Goniometrické funkce a výrazy

Velmi zajímavou kapitolou jsou goniometrické funkce. Jsou to funkce periodické, proto mnohé hodnoty budou vyjádřeny pomocí periody. V programu jsou funkce definované v **R**, to znamená, že proměnná je v radiánech. Pro převod ze stupňů na radiány použijte konstantu Degree.

Kromě základních funkcí $\sin x$, $\cos x$, $\mathrm{tg}\, x$, $\mathrm{cotg}\, x$ jsou definovány funkce sekans $\sec x = \frac{1}{\sin x}$ a kosekans $\mathrm{scs}\, x = \frac{1}{\cos x}$, jejich definiční obory jsou totožné s funkcemi $\mathrm{tg}\, x$, $\mathrm{cotg}\, x$.

Příklad 53

Vytvořme tabulku hodnot funkcí $\sin x$, $\cos x$, $\mathrm{tg}\, x$, $\mathrm{cotg}\, x$ pro úhly 0, $\frac{\pi}{6}$, $\frac{\pi}{4}$, $\frac{\pi}{3}$, $\frac{\pi}{2}$, pokud jsou v těchto hodnotách funkce definované.

Použijeme funkce Table[], která vytvoří sady funkčních hodnot pro daný úhel, uspořádání do tabulek provedeme procedurou TableForm[].

```
In[1]:= Table[{Sin[x], Cos[x], Tan[x], Cot[x]},
          {x, {0, π/6, π/4, π/3, π/2}}] // TableForm
```

Out[1]//TableForm=

0	1	0	ComplexInfinity
$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{\sqrt{3}}$	$\sqrt{3}$
$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	1	1
$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	$\sqrt{3}$	$\frac{1}{\sqrt{3}}$
1	0	ComplexInfinity	0

Obrázek 130 – Příklad 53 – první zobrazení

Do formy tabulky s hlavičkou převedeme tyto údaje pomocí direktivy TableHeadings.

```
In[2]:= TableForm [Table[{Sin[x], Cos[x], Tan[x], Cot[x]},
          {x, {0, π/6, π/4, π/3, π/2}}],
        TableHeadings → {{"0", "π/6", "π/4", "π/3", "π/2"},
          {"Sin x", "Cos x", "Tg x", "Cotg x"}}]
```

Out[2]//TableForm=

	Sin x	Cos x	Tg x	Cotg x
0	0	1	0	ComplexInfin
$\frac{\pi}{6}$	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{\sqrt{3}}$	$\sqrt{3}$
$\frac{\pi}{4}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	1	1
$\frac{\pi}{3}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	$\sqrt{3}$	$\frac{1}{\sqrt{3}}$
$\frac{\pi}{2}$	1	0	ComplexInfinity	0

Obrázek 131 – Příklad 53 – zobrazení s hlavičkou

Příklad 54

Seřaďte podle velikosti tyto funkční hodnoty: $\sin\frac{5\pi}{3}$, $\mathrm{tg}\frac{5\pi}{3}$, $\mathrm{cotg}\frac{5\pi}{3}$, $\cos\frac{5\pi}{3}$.

```
In[1]:= Sin[5 π / 3]
Out[1]= -√3 / 2

In[2]:= Tan[5 π / 3]
Out[2]= -√3

In[3]:= Cot[5 π / 3]
Out[3]= -1 / √3

In[4]:= Cos[5 π / 3]
Out[4]= 1 / 2
```

Obrázek 132 – Příklad 54 – výpočet hodnot

Seřadit hodnoty můžeme funkcí Sort[], zde je nutné díky tvaru výsledných hodnot zadat direktivu Less, aby řazení proběhlo podle hodnoty čísel.

```
In[5]:= Sort[{Sin[5 π / 3], Tan[5 π / 3], Cot[5 π / 3], Cos[5 π / 3]}, Less]
Out[5]= {-√3, -√3 / 2, -1 / √3, 1 / 2}
```

Obrázek 133 – Příklad 54 – seřazení

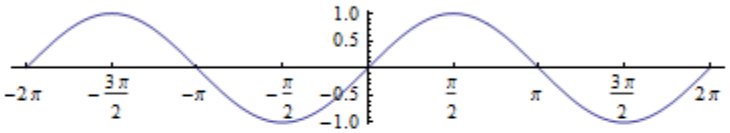
Příklad 55

Zobrazte grafy funkcí $\sin x$, $\cos x$ s násobky $\frac{\pi}{2}$ na ose x.

V tomto případě použijeme funkci Plot[] s direktivou Ticks pro zobrazení vybraných hodnot na ose x a direktivou AspectRatio pro zachování měřítka na obou osách.

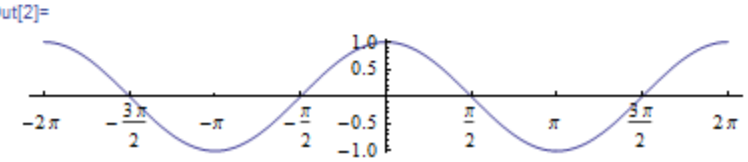
```
In[1]:= Plot[Sin[x], {x, -2 π, 2 π},
          Ticks → {{-2 π, -3 π / 2, -π, -π / 2, 0, π / 2, π, 3 π / 2, 2 π},
            Automatic}, AspectRatio → Automatic]
```

Out[1]=




```
In[2]:= Plot[Cos[x], {x, -2 π, 2 π},
          Ticks → {{-2 π, -3 π / 2, -π, -π / 2, 0, π / 2, π, 3 π / 2, 2 π},
            Automatic}, AspectRatio → Automatic]
```

Out[2]=



Obrázek 134 – Příklad 55

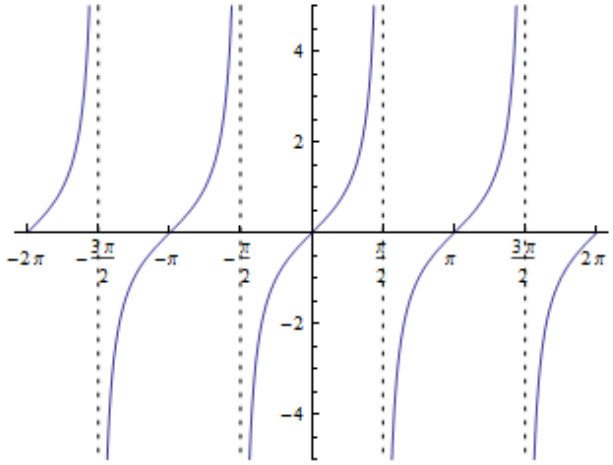
Příklad 56

Zobrazte grafy funkcí tg x, cotg x s násobky $\frac{\pi}{2}$ na ose x.

Postupujeme stejně jako v předchozím příkladu. Direktivou Exclusions a ExclusionsStyle nastavíme svislé asymptoty a jejich zobrazení.

```
In[1]:= Plot[Tan[x], {x, -2 π, 2 π},
        Ticks → {{-2 π, -3 π / 2, -π, -π / 2, 0, π / 2, π, 3 π / 2, 2 π},
          Automatic}, AspectRatio → Automatic,
        PlotRange → {-5, 5}, Exclusions → {Cos[x] == 0},
        ExclusionsStyle → Dashing[Small]]
```

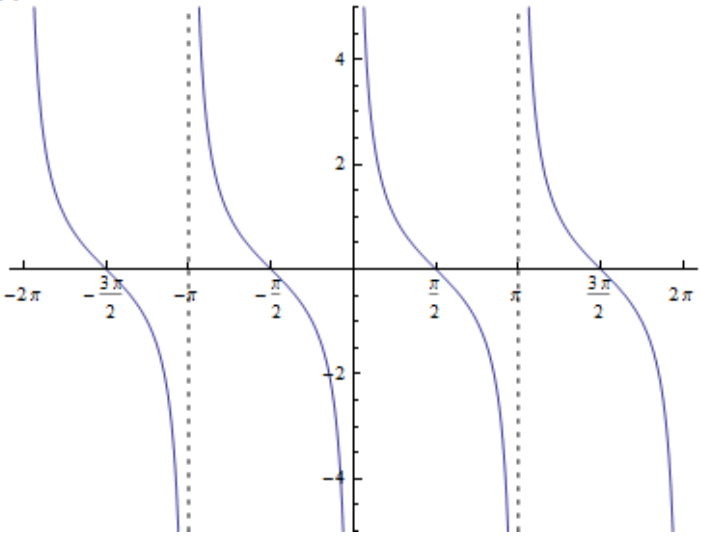
Out[1]=



Obrázek 135 – Příklad 56 – funkce tg *x*

```
In[2]:= Plot[Cot[x], {x, -2 π, 2 π},
        Ticks → {{-2 π, -3 π / 2, -π, -π / 2, 0, π / 2, π, 3 π / 2, 2 π},
          Automatic}, AspectRatio → Automatic,
        PlotRange → {-5, 5}, Exclusions → {Sin[x] == 0},
        ExclusionsStyle → Dashing[Small]]
```

Out[2]=



Obrázek 136 – Příklad 56 – funkce cotg *x*

Pro řešení rovnic použijeme opět funkce Solve[] a Reduce[]. První zobrazí řešení v oboru hodnot funkcí ArcSin[] a ArcCos[], ArcTan[], ArcCot[] podle použitých funkcí. Druhá funkce zobrazí všechna řešení včetně period. Pokud se nám výsledek zobrazí neúplně, využijeme procedury Simplify[] nebo N[].

Příklad 57

Vypočítejme rovnice v **R**:

a) $\cos x = \frac{\sqrt{2}}{2}$, b) $\operatorname{tg}\left(x + \frac{\pi}{3}\right) = 2$.

```
In[1]:= Solve[Cos[x] == Sqrt[2] / 2, x, ]

Solve::bdomv : Warning: Null is not a valid domain
    specification. Mathematica is assuming it is a variable to eliminate. >>

Solve::ifun :
  Inverse functions are being used by Solve, so some solutions may not be found;
    use Reduce for complete solution information. >>

Out[1]=
{{x -> -Pi/4}, {x -> Pi/4}}
```

```
In[2]:= Reduce[Cos[x] == Sqrt[2] / 2, x] // TraditionalForm

Out[2]//TraditionalForm=
c1 ∈ ℤ ⋀ (x = 2 π c1 - π/4 ⋁ x = 2 π c1 + π/4)
```

Obrázek 137 – Příklad 57 – zadání a)

```
In[3]:= Solve[Tan[x + Pi/3] == 2, x]

Solve::ifun :
  Inverse functions are being used by Solve, so some solutions may not be found;
    use Reduce for complete solution information. >>

Out[3]=
{{x -> 1/6 (π - 6 ArcCot[2])}}
```

```
In[4]:= Reduce[Tan[x + Pi/3] == 2, x] // TraditionalForm

Out[4]//TraditionalForm=
c1 ∈ ℤ ⋀ x = -π c1 + π/6 - cot^-1(2)
```

```
In[5]:= x /. {x -> 1/6 (π - 6 ArcCot[2])} // N

Out[5]=
0.0599512
```

Obrázek 138 – Příklad 57 – zadání b)

Výsledek je ovlivněný symbolickým výpočtem. První příklad vypíše řešení v oboru hodnot funkce ArcCos[], lepší zápis dostaneme funkcí Reduce[]. V druhém příkladu je vhodné výsledek v základní periodě číselně vyjádřit, jak ukazuje výpočet s indexem 5.

Příklad 58

Zjednodušte a určete pro která $x \in \mathbf{R}$ má výraz smysl.

a) $\frac{\cos x}{1-\sin x}+\frac{\cos x}{1+\sin x}$, b) $\frac{\operatorname{tg} x}{1+\operatorname{tg}^2 x}$.

```
In[1]:= Cos[x]/(1 - Sin[x]) + Cos[x]/(1 + Sin[x]) // Simplify
Out[1]=
  2 Sec[x]

In[2]:= Reduce[{1 - Sin[x] ≠ 0, 1 + Sin[x] ≠ 0}, x] // Simplify
Out[2]=
  Cos[x] ≠ 0

In[3]:= Reduce[Cos[x] == 0, x] // TraditionalForm
Out[3]//TraditionalForm=
  c1 ∈ ℤ ⋀ (x = 2 π c1 - π/2 ⋁ x = 2 π c1 + π/2)
```

Obrázek 139 – Příklad 58 – zadání a)

První výpočet je nutné převést do standardních funkcí $2\sec x=\frac{2}{\cos x}$. V druhém určíte pro jaké x má výraz smysl. Ten dále zpracujeme, pro hledání řešení lze doporučit výpočet hodnoty x pro rovnost $\cos x=0$. Informace podané programem jsou kompletní.

```
In[4]:= Tan[x]/(1 + (Tan[x])^2) // Simplify
Out[4]=
  Cos[x] Sin[x]

In[5]:= Reduce[{1 + (Tan[x])^2 ≠ 0, cos[x] ≠ 0}, x] // Simplify
Out[5]=
  Sec[x] ≠ 0 && cos[x] ≠ 0

In[6]:= Reduce[Sec[x] == 0, x]
Out[6]=
  False

In[7]:= Reduce[Cos[x] == 0, x] // TraditionalForm
Out[7]//TraditionalForm=
  c1 ∈ ℤ ⋀ (x = 2 π c1 - π/2 ⋁ x = 2 π c1 + π/2)
```

Obrázek 140 – Příklad 58 – zadání b)

V zadání b) je zjednodušení srozumitelné. I určení podmínek platnosti výrazu se jeví jako úplné. Podmínka $\cos x \neq 0$ vyplývá z definiční omezení vstupních funkcí, zde funkce tg x. Další výpočty ukazují, že první omezení je nadbytečné, taková x neexistují (výsledek False) a druhé omezení je shodné s předchozím příkladem.

Příklad 59

Dokažte pravdivost rovnosti $\frac{1+\cos 2x}{\sin 2x} = \text{cotg } x$ a určete, pro která $x \in \mathbf{R}$ má rovnost smysl.

```
In[8]:= (1 + Cos[2 x])/Sin[2 x] == Cot[x]

Out[8]=
(1 + Cos[2 x]) Csc[2 x] == Cot[x]

In[9]:= (1 + Cos[2 x])/Sin[2 x] == Cot[x] // Simplify

Out[9]=
True

In[10]:=
Reduce[{Sin[x] ≠ 0, Sin[2 x] ≠ 0}, x] // Simplify

Out[10]=
Sin[2 x] ≠ 0 && Sin[x] ≠ 0

In[11]:=
Reduce[{Sin[x] == 0 ⋁ Sin[2 x] == 0}, x] // Simplify //
 TraditionalForm

Out[11]//TraditionalForm=
c1 ∈ ℤ ⋀ (x = π c1 ⋁ x = 2 π c1 ⋁ x = π (c1 + 1/2) ⋁ 2 π c1 + π = x)
```

Obrázek 141 – Příklad 59

Pouhý zápis rovnosti výrazů nepostačuje. Potvrzení pravdivosti lze tedy podat použitím funkce Simplify[]. Podmínky platnosti je lepší opět hledat rovností, než nerovností. Výsledek je zapsán složitě, ale lze ho redukovat na podmínku: $x \neq \pi c_1 + \frac{\pi}{2} \wedge c_1 \in \mathbf{Z}$.

5 Komplexní čísla

Při řešení mnohých matematických příkladů nevystačíme s oborem **R**. Jednoduchou ukázkou je řešení rovnice $x^2+1=0$. Ta nemá v oboru **R** žádné řešení, ale v oboru komplexních čísel, která rozšiřují obor **R** o další rozměr. Pro práci s komplexními čísly potřebujeme specifické funkce zaměřené na práci s nimi.

Komplexní číslo zavádíme ve tvaru $z=a+\mathrm{i}\,b$, kde a je reálná složka komplexního čísla, b je imaginární složka komplexního čísla a i je imaginární jednotkou $\mathrm{i}=\sqrt{-1}$. Tento zápis komplexního čísla nazýváme algebraickým tvarem. Kromě tohoto tvaru existuje i goniometrický tvar $z=|z|(\cos\varphi+\mathrm{i}\sin\varphi)$, kde $|z|$ je absolutní hodnota komplexního čísla a φ je argumentem čísla. Stejné je to i pro takzvaný exponenciální tvar $z=|z|\mathrm{e}^{\mathrm{i}\varphi}$. Dalším pojmem je komplexně sdružené číslo $\bar{z}$ k číslu z. Komplexně sdružené číslo má argument s opačným znaménkem a stejnou absolutní hodnotou s původním číslem. Tato čísla se liší pouze znaménkem u imaginární části čísla.

MATHEMATICA nabízí pro práci s komplexními čísly tyto funkce:

Im[]	imaginární část komplexního čísla,
Re[]	reálná část komplexního čísla,
Abs[]	absolutní hodnota (vzdálenost čísla od počátku),
Arg[]	argument komplexního čísla,
Conjugate[]	komplexně sdružené číslo,
Sign[]	jednotkový vektor se stejným argumentem jako zadané číslo,
I	komplexní jednotka $\mathrm{i}=\sqrt{-1}$.

Příklad 60

Vypočítejte hodnotu výrazu $\frac{2+\mathrm{i}}{\mathrm{i}}+\frac{\mathrm{i}}{\mathrm{i}+1}-\frac{2\mathrm{i}+1}{\mathrm{i}-1}$.

```
In[1]:= (2 + I)/I + I/(I + 1) - (2 I + 1)/(I - 1)

Out[1]=
1
```

Obrázek 142 – Příklad 60

Stačí výraz zapsat a přepočítat Kernelem.

Příklad 61

Určete argument a absolutní hodnotu $z_1=1-\mathrm{i}$, najděte číslo komplexně sdružené, vypočítejte z_1^6, vyjádřete argument a absolutní hodnotu této mocniny.

```
In[1]:= z1 = 1 - I

Out[1]=
1 - i

In[2]:= Abs[z1]

Out[2]=
√2

In[3]:= Arg[z1]

Out[3]=
-π/4
```

Obrázek 143 – Příklad 61 – absolutní hodnota a argument čísla

```
In[4]:= Conjugate[z1]

Out[4]=
1 + i

In[5]:= z1^6

Out[5]=
8 i

In[6]:= Abs[z1^6]

Out[6]=
8

In[7]:= Arg[z1^6]

Out[7]=
π/2
```

Obrázek 144 – Příklad 61 – komplexně sdružené číslo a mocnina čísla

Na daném příkladu si ověříme nejen vlastnosti komplexně sdruženého čísla, ale i vlastnosti mocnin komplexních čísel. Absolutní hodnota se umocňuje a argument se při umocňování násobí.

Příklad 62

Vypočítejte rovnice v **C**: a) $3x^2 - 2x + 1 = 0$, b) $2z + 3\overline{z} = 5 + i$.

```
In[1]:= Solve[3 x^2 - 2 x + 1 == 0, x]

Out[1]=
{{x → 1/3 (1 - i √2)}, {x → 1/3 (1 + i √2)}}
```

Obrázek 145 – Příklad 62 – zadání a)

Vyřešit kvadratickou rovnici v **C** není pro program žádný problém, stačí použít osvědčené metody pro řešení rovnic.

```
In[2]:= Solve[2 z + Conjugate[3 z] == 5 + I, z]

Out[2]=
{{z → 1 - i}}
```

Obrázek 146 – Příklad 62 – zadání b)

Ani rovnice s použitím komplexně sdruženého čísla v zadání není pro funkci Solve[] žádný problém. Samozřejmě, že použijeme i další metody Reduce[], NSolve[] a další.

Příklad 63

Vypočítejte všechna řešení rovnice $x^6 + 64 = 0$ v **C**. Výsledky zobrazte v Gaussově rovině.

```
In[1]:= reseni = Solve[x^6 == -64, x]

Out[1]=
{{x → -2 i}, {x → 2 i}, {x → -2 (-1)^(1/6)},
 {x → 2 (-1)^(1/6)}, {x → -2 (-1)^(5/6)}, {x → 2 (-1)^(5/6)}}

In[2]:= Arg[x] /. reseni

Out[2]=
{-π/2, π/2, -5 π/6, π/6, -π/6, 5 π/6}

In[3]:= Abs[x] /. reseni

Out[3]=
{2, 2, 2, 2, 2, 2}
```

Obrázek 147 – Příklad 63 – první výpočet

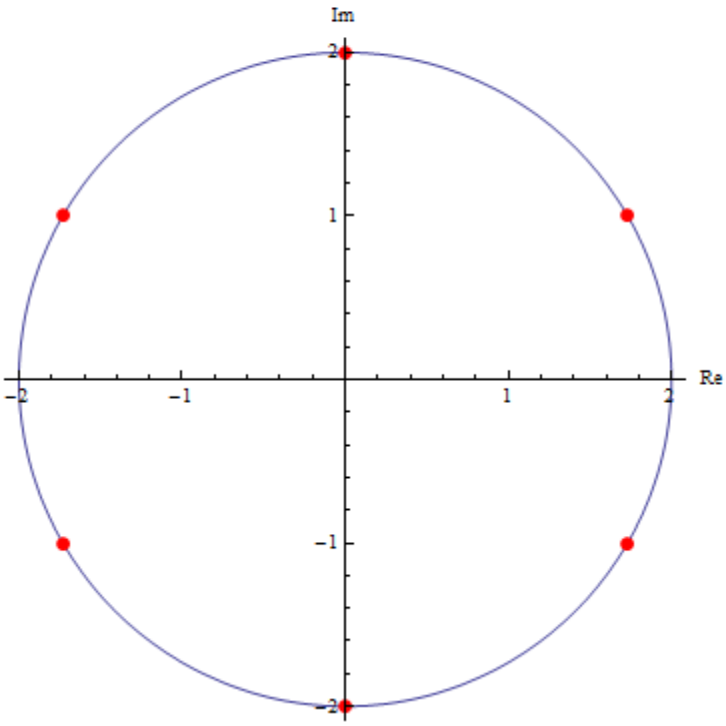
Výsledek funkce Solve[] je v tomto případě málo srozumitelný. Lze vidět, že absolutní hodnota výsledků je 2, a proto výsledky leží na kružnici se středem v počátku o poloměru 2. Argumenty výsledků poukazují na rovnoměrné rozložení řešení na této kružnici. Pokud nám bude stačit číselný výsledek, použijeme metodu NSolve[]. Pro přesný výsledek je vhodné, když vyjádříme reálnou a imaginární složku výsledků v uspořádaných dvojicích.

```
In[4]:= NSolve[x^6 == -64, x] // Simplify

Out[4]=
{{x → -1.73205 - 1. i}, {x → -1.73205 + 1. i}, {x → 0. - 2. i},
 {x → 0. + 2. i}, {x → 1.73205 - 1. i}, {x → 1.73205 + 1. i}}

In[5]:= body = {Re[x], Im[x]} /. reseni

Out[5]=
{{0, -2}, {0, 2}, {-√3, -1}, {√3, 1}, {√3, -1}, {-√3, 1}}
```

Obrázek 148 – Příklad 63 – vyčíslení výsledků

```
In[6]:= graf1 = ListPlot[body, AspectRatio → Automatic,
          PlotStyle → {Red, PointSize[0.02]} ];
        graf2 = ContourPlot[x^2 + y^2 == 2^2, {x, -2, 2}, {y, -2, 2},
          Axes → True, Frame → False, AxesLabel → {"Re", "Im"}];
        Show[graf2, graf1]
```

Out[8]=



Obrázek 149 – Příklad 63 – grafické zobrazení výsledků v Gaussově rovině

Pokud máme komplexní řešení převedené do souřadnic, zakreslíme výsledek v Gaussově rovině. Direktiva AspectRatio nastaví stejné měřítko na obou osách, aby se kružnice zobrazila ve správné podobě, Axes vykreslí osy, Frame naopak zruší zobrazení rámových os. Oba grafy spojí do jednoho funkce Show[].

6 Vektory a analytická geometrie

Vektorem rozumíme množinu stejně velkých a stejně orientovaných úseček. V matematickém zápisu je takový vektor zastoupen uspořádanou *n*-ticí. V programu MATHEMATICA se zapisuje do složených závorek $v = \{v_1, v_2, v_3\}$. Základní operace s vektory jsou:

+	součet vektorů,
-	rozdíl vektorů,
.	skalární součin, lze ho zapsat funkcí Dot[],
Cross[]	vektorový součin,
$v[[i]]$	položka vektoru s indexem *i*,
Norm[]	velikost vektoru,
Normalize[]	normalizovaný vektor,
Length[]	dimenze vektoru,
VectorAngle[]	úhel mezi vektory,
Det[]	determinant čtvercové matice.

Program samozřejmě nabízí i další funkce, orientované na bázové vektory dimenze prostoru a další funkce. Jejich přehled nalezneme v dokumentaci programu v sekci „Operations on Vectors“.

Pro řešení úloh je nutné znát základní pravidla pro práci s vektory a s objekty v analytické geometrii. Následující příklady jsou jen nahlédnutím do analytické geometrie. Nepokrývají celý rozsah, ale ukazují možnosti využití programu MATHEMATICA ve středoškolských úlohách. Ty lze modifikovat a využít při výuce.

Příklad 64

Jsou dány vektory $\mathbf{v} = (1,2,3)$ a $\mathbf{u} = (-2,1,-1)$. Vypočítejte $\mathbf{v}+\mathbf{u}$, $\mathbf{v}-\mathbf{u}$, $2\mathbf{v}$, $\mathbf{v}\cdot\mathbf{u}$, $\mathbf{v}\times\mathbf{u}$, $|\mathbf{v}|$ a úhel mezi vektory.

```
In[1]:= v = {1, 2, 3}
Out[1]=
  {1, 2, 3}

In[2]:= u = {-2, 1, -1}
Out[2]=
  {-2, 1, -1}

In[3]:= v + u
Out[3]=
  {-1, 3, 2}

In[4]:= v - u
Out[4]=
  {3, 1, 4}

In[5]:= 2 v
Out[5]=
  {2, 4, 6}
```

Obrázek 150 – Příklad 64 – součet, rozdíl a násobek vektorů

```
In[6]:= v.u
Out[6]=
  -3

In[7]:= Cross[v, u]
Out[7]=
  {-5, -5, 5}

In[8]:= Norm[v]
Out[8]=
  √14
```

Obrázek 151 – Příklad 64 – skalární součin, vektorový součin a velikost vektoru

Tečkou označujeme skalární součin. Vektorový součin lze zadat také symbolem operace, který nalezneme na paletě Typesetting na záložce Operators pod označením Cross. Klávesová zkratka pro tento symbol je pořadí kláves Esc, cross, Esc.

```
In[9]:= VectorAngle[v, u]
Out[9]=
  ArcCos[-√(3/7)/2]

In[10]:=
  VectorAngle[v, u] / Degree // N
Out[10]=
  109.107
```

Obrázek 152 – Příklad 64 – úhel mezi vektory

Výsledek funkce je udán v radiánech. Díky symbolickým výpočtům je výsledek těžko srozumitelný, proto zvolíme numerické vyjádření. Vydělením výsledku převodovou konstantou Degree, převedeme výsledek z radiánů na stupně.

Příklad 65

Je dán vektor $\mathbf{u} = \left(\sqrt{3}, 1\right)$, určete souřadnice vektoru $\mathbf{v}$, jehož velikost je 4 a svírá s vektorem $\mathbf{u}$ úhel 60°.

Funkce VectorAngle[] není vhodná do výpočtů a rovnic. Proto si připomeneme vztah mezi skalárním součinem a úhlem dvou vektorů: $\mathbf{v} \cdot \mathbf{u} = |\mathbf{v}|\,|\mathbf{u}| \cos\varphi$. Společně s velikostí hledaného vektoru $|\mathbf{v}| = 4$ dostáváme dvě rovnice s neznámými souřadnicemi vektoru $\mathbf{v}$. Takovou soustavu vyřešíme metodou Solve[]. Pro správnou funkci výpočtu je nutné předem definovat $\mathbf{v}$ jako dvojrozměrný vektor.

```
In[1]:= u = {√3 , 1};
        v = {a, b};

In[3]:= Solve[{u.v == Norm[u] 4 Cos[60 Degree], Norm[v] == 4}, v]
Out[3]=
        {{a → 2 √3 , b → -2}, {a → 0, b → 4}}
```

Obrázek 153 – Příklad 65 – výpočet

Výsledkem jsou dva možné vektory $\mathbf{v}_1 = (2\sqrt{3},-2)$, $\mathbf{v}_1 = (0,4)$. Podobným postupem k danému vektoru najdeme kolmý vektor dané velikosti.

Příklad 66

Určete, zda jsou vektory $\mathbf{a} = (2,-1,3)$, $\mathbf{b} = (3,0,6)$, $\mathbf{c} = (7,-5,10)$ lineárně závislé.

Z definice lineární nezávislosti vektorů plyne, že dané vektory jsou lineárně nezávislé tehdy a jen tehdy pokud existuje pouze triviální řešení (k_1,k_2,k_3) rovnice $k_1\mathbf{a} + k_2\mathbf{b} + k_3\mathbf{c} = (0,0,0)$. Pod pojmem triviální řešení rozumíme, že všechny k_i jsou rovny nule.

```
In[1]:= a = {2, -1, 3};
        b = {3, 0, 6};
        c = {7, -5, 10};
        Solve[a * k1 + b * k2 + c * k3 == {0, 0, 0}, {k1, k2, k3}]
Out[4]=
        {{k1 → 0, k2 → 0, k3 → 0}}
```

Obrázek 154 – Příklad 66 – lineární nezávislost vektorů

V našem případě jsou vektory lineárně nezávislé. Podobný postup zvolíme pro určení lineární kombinace vektorů.

Příklad 67

Přímka *p* je zadána bodem $A = [1,2]$ a směrovým vektorem $\mathbf{s} = (-2,3)$. Zapište její parametrický a obecný tvar.

```
In[1]:= Clear[s, c, t, cc, x, y, n, bodA, primkaPar, primkaOb]
        bodA = {1, 2};
        s = {-2, 3};
        primkaPar = bodA + s * t
        n = {3, 2};
        cc = Solve[n[[1]] * bodA[[1]] + n[[2]] * bodA[[2]] + c == 0,
            c];
        primkaOb = n[[1]] x + n[[2]] y + c == 0 /. cc
Out[4]=
        {1 - 2 t, 2 + 3 t}

Out[7]=
        {-7 + 3 x + 2 y == 0}
```

Obrázek 155 – Příklad 67 – přímka

Aby nedošlo k chybě ve výpočtu, je vhodné všechny použité proměnné uvolnit funkcí Clear[]. Zapsat parametrický zápis přímky je jednoduché. Na obecný tvar ale potřebujeme normálový vektor přímky. Ten můžeme získat například postupem z příkladu 65. Parametr *c* vypočítáme dosazením bodu A do prototypu rovnice s dosazenými parametry a a b z normálového vektoru. Výsledek je v tomto příkladu uložen v proměnné *cc*.

Příklad 68

Určete vzájemnou polohu přímky $p = \{[1+2t, 2-3t], t \in \mathbf{R}\}$ a $q : 2x + y - 1 = 0$.

Vzájemnou polohu určíme z množství společných bodů. Vyřešíme tedy soustavu rovnic.

```
In[1]:= Clear[x, y, t]
        Solve[{x == 1 + 2 t, y == 2 - 3 t, 2 x + y - 1 == 0}, {x, y, t}]

Out[2]=
  {{x → -5, y → 11, t → -3}}
```

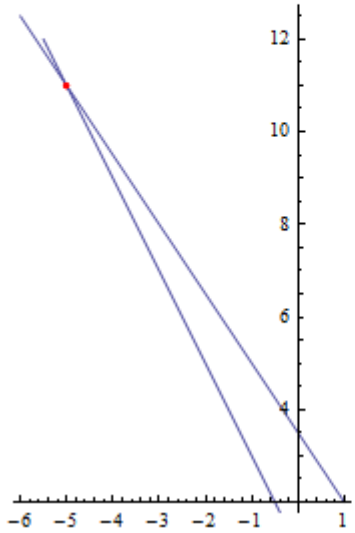
Obrázek 156 – Příklad 68 – společný bod

Výsledkem je jediný bod $P = [-5,11]$, to znamená, že se jedná o různoběžky. V zadání funkce Solve[] jsme museli zadat rovnice parametrické přímky *p* a obecnou rovnici *q*.

Výsledek se dá zobrazit v grafu.

```
In[3]:= grafp = ParametricPlot[{1 + 2 t, 2 - 3 t}, {t, -3.5, 0}];
        grafq = Plot[1 - 2 x, {x, -5.5, 0}];
        grafbod = ListPlot[{{-5, 11}},
           PlotStyle → {Red, PointSize[0.02]}];
        Show[grafp, grafq, grafbod]
```

Out[6]=



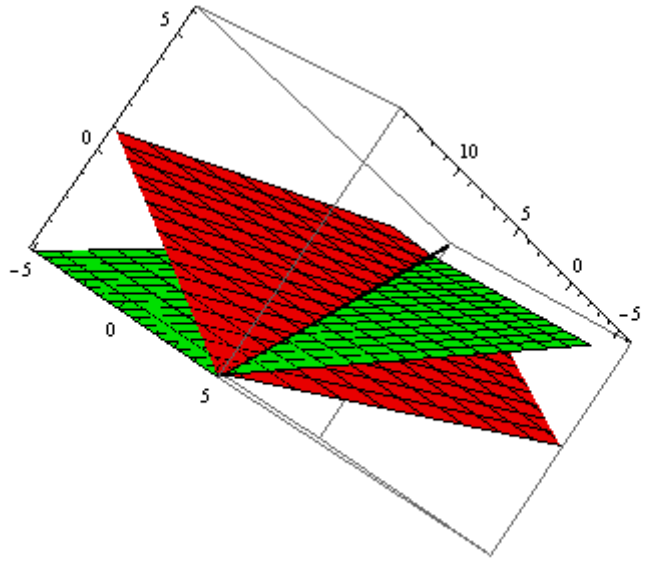
Obrázek 157 – Příklad 68 – grafické řešení

Příklad 69

Zobrazte roviny $\rho : 2x + 4y + z - 8 = 0$ a $\delta : 2y + z - 6 = 0$ v 3D grafu a určete vzájemnou polohu rovin.

I zde je důležité najít společné body (přímku).

```
In[1]:= Solve[{2 x + 4 y + z - 8 == 0, 2 y + z - 6 == 0}, {x, y, z}]
  Solve::svars : Equations may not give solutions for all "solve" variables. >>
Out[1]=
  {{y → 1 - x, z → 4 + 2 x}}

In[2]:= primka = {t, 1 - t, 4 + 2 t};
        grafprimka = ParametricPlot3D[primka, {t, -5, 5},
           PlotStyle → Thick];
        grafrovina = Plot3D[{8 - 2 x - 4 y, 6 - 2 y}, {x, -5, 5},
           {y, -5, 5}, PlotStyle → {Red, Green}];
        Show[grafprimka, grafrovina]
Out[5]=
```



Obrázek 158 – Příklad 69

Z výsledků funkce Solve[] dostaneme množinu společných bodů. V tomto případě tyto body tvoří přímku s parametrem, který odpovídá souřadnici *x*. Výsledek je přepsán do proměnné *primka*. Z obrázku rychle poznáte vzájemnou polohu rovin.

Analytická geometrie je rozsáhlá, ale na všechny analytické úlohy nalezneme nástroje v programu MATHEMATICA. Výhodou programu je i názorný grafický výstup, který nám pomůže při vytvoření představy o objektech, se kterými pracujeme.

7 Posloupnosti a kombinatorika

Tato kapitola spojuje dvě rozdílné části matematiky, proto ji rozdělíme na dvě samostatné části. Dostali se do jedné stati díky svému rozsahu.

7.1 Kombinatorika

Program obsahuje asi 450 funkcí z oboru kombinatoriky. Nalezneme je v dokumentaci pod pojmem „Combinatorica“. Před zadáním některých funkcí musíme aktivovat příkazem <<Combinatorica` knihovnu kombinatoriky.

Přehled základních fukcí:

$n!$	faktoriál n,
Binomial[n, k]	vypočítá hodnotu kombinačního čísla $\binom{n}{k}$,
Permutations[soubor]	sestaví všechny permutace ze souboru (seznamu),
KSubsets[soubor, n]	vytvoří ze souboru n-prvkové podmnožiny souboru,
Length[soubor]	udává počet prvků v souboru,
Dimensions[matice]	určí rozměry matice,
Flatten[matice, n]	sníží dimenzi matice o n řádů,
Outer[List, soubor1, soubor2]	vygeneruje všechny uspořádané dvojice z prvků souborů,
Select[soubor, podmínka]	vybere ze seznamu prvky, které splňují podmínku.

Příklad 70

Sestavte funkce pro výpočet variací bez opakování i s opakováním a kombinací bez opakování i s.

Tato úloha je jednoduchá stačí podle tabulkových vzorců doplnit.

```
In[1]:= var[n_, k_] = n! / (n - k)! // TraditionalForm
```

Out[1]//TraditionalForm=

$$\frac{n!}{(n-k)!}$$

```
In[2]:= varOp[n_, k_] = n^k // TraditionalForm
```

Out[2]//TraditionalForm=

$$n^k$$

```
In[3]:= com[n_, k_] = Binomial[n, k] // TraditionalForm
```

Out[3]//TraditionalForm=

$$\binom{n}{k}$$

```
In[4]:= comOp[n_, k_] = Binomial[n + k - 1, k] // TraditionalForm
```

Out[4]//TraditionalForm=

$$\binom{k+n-1}{k}$$

Obrázek 159 – Příklad 70

Příklad 71

Vyřešte rovni rovnici $\binom{x}{x-1}+\binom{x-1}{x-3}=4$.

```
In[1]:= Solve[Binomial[x, x - 1] + Binomial[x - 1, x - 3] == 4, x, Integers]

Out[1]=
  {{x → -2}, {x → 3}}
```

Obrázek 160 – Příklad 71

Podmínkám zadání vyhovuje pouze jeden výsledek a to 3.

Příklad 72

Sestavte všechny tříprvkové podmnožiny množiny $\{A, B, C, D\}$ a určete jejich počet.

```
In[1]:= << Combinatorica`

  General::compat :
    Combinatorica Graph and Permutations functionality has been superseded by preloaded
      functionaliy. The package now being loaded may conflict
      with this. Please see the Compatibility Guide for details.

In[2]:=
      c = KSubsets[{A, B, C, D}, 3]

Out[2]=
  {{A, B, C}, {A, B, D}, {A, C, D}, {B, C, D}}

In[3]:= Length[c]

Out[3]=
  4
```

Obrázek 161 – Příklad 72

Prvním příkazem aktivujeme knihovnu kombinatoriky, ve které je funkce KSubsets[].

Příklad 73

Zjistěte kolik tříciferných čísel lze sestavit z cifer 0,1,2,3,4, pokud se cifry mohou opakovat.

V tomto příkladu se pokusíme všechny takové trojce najít a pak určit jejich počet. Na pouhé zjištění počtu by nám stačilo použít vzorce z příkladu 70. Nejprve vytvoříme všechny uspořádané trojce sestavené z daných cifer funkcí Outer[].

```
In[1]:= m = {0, 1, 2, 3, 4};
        seznam = Outer[List, m, m, m]

Out[2]=
{{{{0, 0, 0}, {0, 0, 1}, {0, 0, 2}, {0, 0, 3}, {0, 0, 4}},
  {{0, 1, 0}, {0, 1, 1}, {0, 1, 2}, {0, 1, 3}, {0, 1, 4}},
  {{0, 2, 0}, {0, 2, 1}, {0, 2, 2}, {0, 2, 3}, {0, 2, 4}},
  {{0, 3, 0}, {0, 3, 1}, {0, 3, 2}, {0, 3, 3}, {0, 3, 4}},
  {{0, 4, 0}, {0, 4, 1}, {0, 4, 2}, {0, 4, 3}, {0, 4, 4}}},
 {{{1, 0, 0}, {1, 0, 1}, {1, 0, 2}, {1, 0, 3}, {1, 0, 4}},
  {{1, 1, 0}, {1, 1, 1}, {1, 1, 2}, {1, 1, 3}, {1, 1, 4}},
  {{1, 2, 0}, {1, 2, 1}, {1, 2, 2}, {1, 2, 3}, {1, 2, 4}},
  {{1, 3, 0}, {1, 3, 1}, {1, 3, 2}, {1, 3, 3}, {1, 3, 4}},
  {{1, 4, 0}, {1, 4, 1}, {1, 4, 2}, {1, 4, 3}, {1, 4, 4}}},
 {{{2, 0, 0}, {2, 0, 1}, {2, 0, 2}, {2, 0, 3}, {2, 0, 4}},
  {{2, 1, 0}, {2, 1, 1}, {2, 1, 2}, {2, 1, 3}, {2, 1, 4}},
  {{2, 2, 0}, {2, 2, 1}, {2, 2, 2}, {2, 2, 3}, {2, 2, 4}},
  {{2, 3, 0}, {2, 3, 1}, {2, 3, 2}, {2, 3, 3}, {2, 3, 4}},
  {{2, 4, 0}, {2, 4, 1}, {2, 4, 2}, {2, 4, 3}, {2, 4, 4}}},
 {{{3, 0, 0}, {3, 0, 1}, {3, 0, 2}, {3, 0, 3}, {3, 0, 4}},
  {{3, 1, 0}, {3, 1, 1}, {3, 1, 2}, {3, 1, 3}, {3, 1, 4}},
  {{3, 2, 0}, {3, 2, 1}, {3, 2, 2}, {3, 2, 3}, {3, 2, 4}},
  {{3, 3, 0}, {3, 3, 1}, {3, 3, 2}, {3, 3, 3}, {3, 3, 4}},
  {{3, 4, 0}, {3, 4, 1}, {3, 4, 2}, {3, 4, 3}, {3, 4, 4}}},
 {{{4, 0, 0}, {4, 0, 1}, {4, 0, 2}, {4, 0, 3}, {4, 0, 4}},
  {{4, 1, 0}, {4, 1, 1}, {4, 1, 2}, {4, 1, 3}, {4, 1, 4}},
  {{4, 2, 0}, {4, 2, 1}, {4, 2, 2}, {4, 2, 3}, {4, 2, 4}},
  {{4, 3, 0}, {4, 3, 1}, {4, 3, 2}, {4, 3, 3}, {4, 3, 4}},
  {{4, 4, 0}, {4, 4, 1}, {4, 4, 2}, {4, 4, 3}, {4, 4, 4}}}}
```

Obrázek 162 – Příklad 73 – vytvoření seznamu všech možných uspořádaných trojic

```
In[3]:= Dimensions[seznam]

Out[3]=
  {5, 5, 5, 3}

In[4]:= seznam1 = Flatten[seznam, 2];
        vysledek = Select[seznam1, #[[1]] > 0 &]
        Length[vysledek]

Out[5]=
  {{1, 0, 0}, {1, 0, 1}, {1, 0, 2}, {1, 0, 3}, {1, 0, 4}, {1, 1, 0},
   {1, 1, 1}, {1, 1, 2}, {1, 1, 3}, {1, 1, 4}, {1, 2, 0}, {1, 2, 1},
   {1, 2, 2}, {1, 2, 3}, {1, 2, 4}, {1, 3, 0}, {1, 3, 1}, {1, 3, 2},
   {1, 3, 3}, {1, 3, 4}, {1, 4, 0}, {1, 4, 1}, {1, 4, 2}, {1, 4, 3},
   {1, 4, 4}, {2, 0, 0}, {2, 0, 1}, {2, 0, 2}, {2, 0, 3}, {2, 0, 4},
   {2, 1, 0}, {2, 1, 1}, {2, 1, 2}, {2, 1, 3}, {2, 1, 4}, {2, 2, 0}, {2, 2, 1},
   {2, 2, 2}, {2, 2, 3}, {2, 2, 4}, {2, 3, 0}, {2, 3, 1}, {2, 3, 2}, {2, 3, 3},
   {2, 3, 4}, {2, 4, 0}, {2, 4, 1}, {2, 4, 2}, {2, 4, 3}, {2, 4, 4}, {3, 0, 0},
   {3, 0, 1}, {3, 0, 2}, {3, 0, 3}, {3, 0, 4}, {3, 1, 0}, {3, 1, 1}, {3, 1, 2},
   {3, 1, 3}, {3, 1, 4}, {3, 2, 0}, {3, 2, 1}, {3, 2, 2}, {3, 2, 3}, {3, 2, 4},
   {3, 3, 0}, {3, 3, 1}, {3, 3, 2}, {3, 3, 3}, {3, 3, 4}, {3, 4, 0}, {3, 4, 1},
   {3, 4, 2}, {3, 4, 3}, {3, 4, 4}, {4, 0, 0}, {4, 0, 1}, {4, 0, 2}, {4, 0, 3},
   {4, 0, 4}, {4, 1, 0}, {4, 1, 1}, {4, 1, 2}, {4, 1, 3}, {4, 1, 4}, {4, 2, 0},
   {4, 2, 1}, {4, 2, 2}, {4, 2, 3}, {4, 2, 4}, {4, 3, 0}, {4, 3, 1}, {4, 3, 2},
   {4, 3, 3}, {4, 3, 4}, {4, 4, 0}, {4, 4, 1}, {4, 4, 2}, {4, 4, 3}, {4, 4, 4}}

Out[6]=
  100
```

Obrázek 163 – Příklad 73 – výběr

Příkazem Flatten[] snížíme dimenzi výsledku na seznam. Vybereme ty, co nezačínají na 0 metodou Select[]. Výsledný počet určíme metodou Length[].

Příklad 74

Určete kolik tříciferných čísel lze sestavit z cifer 0,1,2,3,4, pokud se cifry nemohou opakovat.

Postup je podobný. Využijeme proceduru Permutations[] pro generování pořadí tří prvků. Takto vzniklý seznam už nemusíme dál upravovat. Teď jen vybereme ty permutace, které nezačínají nulou. Výsledný počet opět určíme metodou Length[].

```
In[1]:= m = {0, 1, 2, 3, 4};
        seznam = Permutations[m, {3}]

Out[2]=
  {{0, 1, 2}, {0, 1, 3}, {0, 1, 4}, {0, 2, 1}, {0, 2, 3}, {0, 2, 4},
   {0, 3, 1}, {0, 3, 2}, {0, 3, 4}, {0, 4, 1}, {0, 4, 2}, {0, 4, 3},
   {1, 0, 2}, {1, 0, 3}, {1, 0, 4}, {1, 2, 0}, {1, 2, 3}, {1, 2, 4},
   {1, 3, 0}, {1, 3, 2}, {1, 3, 4}, {1, 4, 0}, {1, 4, 2}, {1, 4, 3}, {2, 0, 1},
   {2, 0, 3}, {2, 0, 4}, {2, 1, 0}, {2, 1, 3}, {2, 1, 4}, {2, 3, 0}, {2, 3, 1},
   {2, 3, 4}, {2, 4, 0}, {2, 4, 1}, {2, 4, 3}, {3, 0, 1}, {3, 0, 2}, {3, 0, 4},
   {3, 1, 0}, {3, 1, 2}, {3, 1, 4}, {3, 2, 0}, {3, 2, 1}, {3, 2, 4}, {3, 4, 0},
   {3, 4, 1}, {3, 4, 2}, {4, 0, 1}, {4, 0, 2}, {4, 0, 3}, {4, 1, 0}, {4, 1, 2},
   {4, 1, 3}, {4, 2, 0}, {4, 2, 1}, {4, 2, 3}, {4, 3, 0}, {4, 3, 1}, {4, 3, 2}}
```

Obrázek 164 – Příklad 74 – seznam permutací

```
In[3]:= vysledek = Select[seznam, #[[1]] > 0 &]
        Length[vysledek]

Out[3]=
  {{1, 0, 2}, {1, 0, 3}, {1, 0, 4}, {1, 2, 0}, {1, 2, 3}, {1, 2, 4},
   {1, 3, 0}, {1, 3, 2}, {1, 3, 4}, {1, 4, 0}, {1, 4, 2}, {1, 4, 3}, {2, 0, 1},
   {2, 0, 3}, {2, 0, 4}, {2, 1, 0}, {2, 1, 3}, {2, 1, 4}, {2, 3, 0}, {2, 3, 1},
   {2, 3, 4}, {2, 4, 0}, {2, 4, 1}, {2, 4, 3}, {3, 0, 1}, {3, 0, 2}, {3, 0, 4},
   {3, 1, 0}, {3, 1, 2}, {3, 1, 4}, {3, 2, 0}, {3, 2, 1}, {3, 2, 4}, {3, 4, 0},
   {3, 4, 1}, {3, 4, 2}, {4, 0, 1}, {4, 0, 2}, {4, 0, 3}, {4, 1, 0}, {4, 1, 2},
   {4, 1, 3}, {4, 2, 0}, {4, 2, 1}, {4, 2, 3}, {4, 3, 0}, {4, 3, 1}, {4, 3, 2}}

Out[4]=
  48
```

Obrázek 165 – Příklad 74 – výsledek

7.2 Posloupnosti

Pod pojmem posloupnost rozumíme reálnou funkci jejíž definičním oborem jsou přirozená čísla, nebo interval přirozených čísel. Hodnotu posloupnosti pro *n* nazveme *n*-tým členem posloupnosti a značíme ho a_n.

V této kapitole si ukážeme užití dalších funkcí programu MATHEMATICA, které se dají použít i v jiných kapitolách matematiky.

Print[výraz]	zobrazí výraz,
Do[příkaz, {*i, max*}]	cyklus vykonávající příkaz s řídící proměnnou *i* od 1 do *max*,
While[podmínka, příkaz]	cyklus vykonávající příkaz dokud je podmínka platná,
For[počáteční podmínky, podmínka, přírůstek, příkaz]	cyklus vykonávající příkaz s řídící proměnnou a daným přírůstkem za splnění podmínky,
Sum[výraz,{*i*, *max*}]	součet výrazů pro *i* od 1 do *max*,
Limit[výraz, $n \to$ hodnota]	limita výrazu pro zadanou hodnotu proměnné *n*.

V následujících příkladech si ukážeme, jak lze vypsat několik členů posloupnosti za daných podmínek.

Příklad 75

Vypište prvních sedm členů posloupnosti $\left(\frac{n+2}{n+1}\right)_{n=1}^{\infty}$.

Z důvodu úspory místa jsou výsledky příkazu Do[] a Table[] vedle sebe. My dostanete tyto výsledky pod sebou. Zároveň vidíme rozdíl mezi výsledkem obou příkazů. Výsledek z Table[] dále zpracováváme například příkazem ListPlot[].

```
In[1]:= a[n_] = (n + 2)/(n + 1);
        Do[Print[a[n]], {n, 7}]
```

$\frac{3}{2}$

$\frac{4}{3}$

$\frac{5}{4}$

$\frac{6}{5}$

$\frac{7}{6}$

$\frac{8}{7}$

$\frac{9}{8}$

```
In[3]:= Table[a[n], {n, 1, 7}]
Out[3]= {3/2, 4/3, 5/4, 6/5, 7/6, 8/7, 9/8}
```

Obrázek 166 – Příklad 75

Příklad 76

Vypište první členy posloupnosti $\left(2^{n-1}\right)_{n=1}^{\infty}$, které jsou menší než 50.

Pokud chceme použít více příkazů, musíme je oddělit středníkem. V tomto případě se jedná o příkaz Print[] a příkaz zvýšení hodnoty řídící proměnné ve tvaru n++.

```
In[1]:= a[n_] = 2^(n-1); n = 1;
        While[a[n] < 50, Print[a[n]]; n++]
```

1

2

4

8

16

32

Obrázek 167 – Příklad 76

Příklad 77

Vypište prvních sedm členů rekurentně zadané posloupnosti:

$a_1 = 1$, $a_2 = 1$; $a_{n+2} = a_{n+1} + a_n$.

Výsledek lze získat libovolným cyklem. V tomto případě jsme zvolili cyklus For[] a odloženou definici funkce.

```
In[1]:= a[n_] := 1 /; n < 3
        a[n_] := a[n - 1] + a[n - 2] /; n ≥ 3

In[3]:= For[n = 1, n < 8, n++, Print[a[n]]]
  1
  1
  2
  3
  5
  8
  13
```

Obrázek 168 – Příklad 77

Příklad 78

Vypočítejte součet prvních dvaceti členů posloupnosti $(2n+1)_{n=1}^{\infty}$.

Z důvodu označení předchozích výpočtů je vhodné nejprve použité proměnné odstranit funkcí Clear[]. Pro jistotu si ještě vypíšeme jednotlivé členy. Samotný výpočet za nás obstará příkaz Sum[].

```
In[1]:= Clear[a, n]

In[2]:= a[n_] = 2 n + 1;
        Sum[a[n], {n, 20}]
Out[3]=
  440

In[4]:= Table[a[n], {n, 1, 20}]
Out[4]=
  {3, 5, 7, 9, 11, 13, 15, 17, 19,
   21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41}
```

Obrázek 169 – Příklad 78

Příklad 79

Dokažte, že je posloupnost $\left(\frac{2n}{n+1}\right)_{n=1}^{\infty}$ rostoucí a shora omezená.

Vypočítejte limitu posloupnosti v ∞.

```
In[1]:= a[n_] = 2 n / (n + 1);
        Table[a[n], {n, 1, 10}]
Out[2]= {1, 4/3, 3/2, 8/5, 5/3, 12/7, 7/4, 16/9, 9/5, 20/11}
```

Obrázek 170 – Příklad 79 – ukázka posloupnosti

Důkaz, že posloupnost je rostoucí, provedeme funkcí Reduce[], kde zjistíme, pro jaké *n* platí nerovnost $a_{n+1} > a_n$.

```
In[3]:= Reduce[{a[n + 1] > a[n], n ≥ 1}, n, Integers]
Out[3]= n ∈ Integers && n ≥ 1
```

Obrázek 171 – Příklad 79 – důkaz rostoucí posloupnosti

Podobně provedeme důkaz omezenosti posloupnosti hodnotou 2.

```
In[4]:= Reduce[{a[n] < 2, n ≥ 1}, n, Integers]
Out[4]= n ∈ Integers && n ≥ 1
```

Obrázek 172 – Příklad 79 – důkaz, že posloupnost je shora omezená

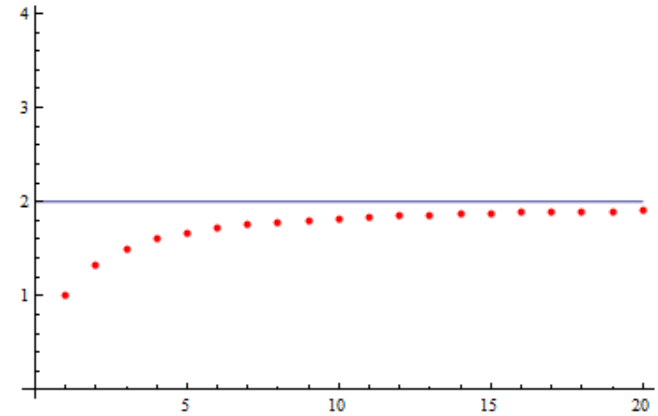
```
In[5]:= Limit[a[n], n → Infinity]
Out[5]= 2
```

Obrázek 173 – Příklad 79 – výpočet limity

Zobrazení rostoucí shora omezené posloupnosti provedeme funkcí Show[], kde spojíme dva grafy. První graf zakreslený metodou Plot[] je horní závorou posloupnosti. Druhý graf je bodový, vytvořený funkcí ListPlot[]. Ten je sestrojen na základě tabulky Table[] prvními dvaceti hodnotami posloupnosti.

```
In[6]:= Show[Plot[2, {x, 0, 20}],
         ListPlot[Table[a[n], {n, 1, 20}],
          PlotStyle → {Red, PointSize[0.012]}]]
```

Out[6]=



Obrázek 174 – Příklad 79 – zobrazení rostoucí shora omezené posloupnosti

8 Diferenciální počet a integrální počet

Program MATHEMATICA nabízí dostatečné nástroje i pro tuto kapitolu. Možnosti využití jsou velmi rozsáhlé a to nejen pro středoškolskou matematiku, ale i pro vysokoškolské užití v praktických i symbolických výpočtech včetně práce s funkcemi více proměnných. Ukážeme si pro jednoduchost jen středoškolské užití na vyšetření zadané funkce a výpočet neurčitého a určitého integrálu. I při použití těchto nástrojů musí studenti chápat pojmy a postupy této kapitoly matematiky.

V následujících ukázkách použijeme funkce:

D[funkce, proměnná]	výpočet derivace,
Integrate[funkce, proměnná]	výpočet integrálu,
Integrate[funkce, {proměnná, dolní mez, horní mez}]	výpočet určitého integrálu,
Limit[výraz, proměnná→mez]	výpočet limity.

Příklad 80

Vyšetřete funkci $y = \frac{x^2}{x+2}$.

Prvním krokem je určení definičního oboru. V tomto příkladě se jedná pouze o jediné omezení $x + 2 \neq 0$.

```
In[1]:= Reduce[x + 2 ≠ 0, x] // TraditionalForm

Out[1]//TraditionalForm=
x + 2 ≠ 0
```

Obrázek 175 – Příklad 80 – definiční obor

Dalším krokem jsou limity v krajích definičního oboru a asymptoty funkce v $\pm\infty$.

```
In[2]:= fce[x_] = x^2/(x + 2);
        k1 = Limit[fce[x] / x, x -> -∞]; q1 = Limit[fce[x] - k1 x, x → +∞];
        k2 = Limit[fce[x] / x, x → +∞]; q2 = Limit[fce[x] - k2 x, x → -∞];
        k1 x + q1
        k2 x + q2
        Limit[fce[x], x → -2, Direction → -1]
        Limit[fce[x], x → -2, Direction → 1]

Out[5]=
-2 + x

Out[6]=
-2 + x

Out[7]=
∞

Out[8]=
-∞
```

Obrázek 176 – Příklad 80 – limity a asymptoty

Nyní pomocí první derivace nalezneme body podezřelé z extrémů a intervaly monotonie funkce. Výstupy jsou zformovány funkcí TraditionalForm[]. Výpočty provedeme příkazy Solve[] a Reduce[]. Metoda Together[] nám zlomky ve výsledku spojí do jednoho. Je to jenom proto, abychom viděli výsledek v podobě, jak jsme zvyklí.

```
In[9]:= fce[x_] = x^2/(x + 2);
        der1[x_] = D[fce[x], x];

        "První derivace: y'= " TraditionalForm[Together[der1[x]]]
        "Body podezřelé  z extrému: " Solve[der1[x] == 0, x, Reals]
        "Funkce je rostoucí pro x: " Reduce[der1[x] > 0, x, Reals] // TraditionalForm
        "Funkce je klesající pro x: " Reduce[der1[x] < 0, x, Reals] // TraditionalForm
```

Out[11]=

$$\text{První derivace: } y' = \frac{x^2 + 4x}{(x+2)^2}$$

Out[12]=

```
{{Body podezřelé  z extrému:  (x → -4)}, {Body podezřelé  z extrému:  (x → 0)}}
```

Out[13]//TraditionalForm=

Funkce je rostoucí pro x: $(x < -4 \vee x > 0)$

Out[14]//TraditionalForm=

Funkce je klesající pro x: $(-4 < x < -2 \vee -2 < x < 0)$

Obrázek 177 – Příklad 80 – první derivace

Teď je na řadě druhá derivace s určením inflexních bodů a intervalů konvexnosti a konkávnosti funkce.

```
In[15]:=
        fce[x_] = x^2/(x + 2);
        der1[x_] = D[fce[x], x];
        der2[x_] = D[der1[x], x];
        "Druhá derivace: y''= " TraditionalForm[Together[der2[x]]]
        "Body podezřelé z inflexe: " Solve[der2[x] == 0, x, Reals] // TraditionalForm
        "Funkce je konvexní pro x: " Reduce[der2[x] > 0, x, Reals] // TraditionalForm
        "Funkce je konkávní pro x: " Reduce[der2[x] < 0, x, Reals] // TraditionalForm
```

Out[18]=

$$\text{Druhá derivace: } y'' = \frac{8}{(x+2)^3}$$

Out[19]//TraditionalForm=

{}

Out[20]//TraditionalForm=

Funkce je konvexní pro x: $(x > -2)$

Out[21]//TraditionalForm=

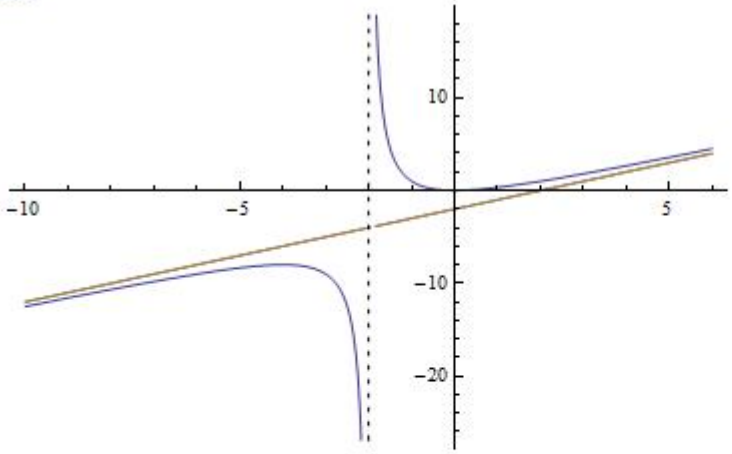
Funkce je konkávní pro x: $(x < -2)$

Obrázek 178 – Příklad 80 – druhá derivace

In[22]:=

```
fce[x_] = x^2/(x + 2);
k1 = Limit[fce[x]/x, x -> -∞]; q1 = Limit[fce[x] - k1 x, x → +∞];
k2 = Limit[fce[x] / x, x → +∞]; q2 = Limit[fce[x] - k2 x, x → -∞];
Plot[{fce[x], k1 x + q1, k2 x + q2}, {x, -10, 6}, Exclusions → {x == -2},
 ExclusionsStyle → Dashing[Small]]
"Průsečíky s osou x: " TraditionalForm[Solve[fce[x] == 0, x, Reals]]
"Průsečíky s osou y: " Rationalize[fce[0]]
```

Out[25]=



Out[26]=

```
Průsečíky s osou x: {{x → 0}, {x → 0}}
```

Out[27]=

```
0
```

Obrázek 179 – Příklad 80 – graf

```
fce[x_] = x^2/(x + 2);
TableForm [Table[{x, fce[x]}, {x, {-4, 0}}], TableHeadings → {None, {"x", "y"}}]
```

x	y
-4	-8
0	0

Obrázek 180 – Příklad 80 – tabulka

Z těchto výsledků lze příkazy Print[] sestavit přehledný grafický výstup. Zadání a výsledek si prohlédneme na následujících stránkách.

```
In[1]:= fce[x_] = x^2/(x + 2);
        k1 = Limit[fce[x]/x, x -> -∞]; q1 = Limit[fce[x] - k1 x, x → +∞];
        k2 = Limit[fce[x] / x, x → +∞]; q2 = Limit[fce[x] - k2 x, x → -∞];
        Print[" Lim x→-2+ ", Text[ fce[x]], " = ", Limit[fce[x], x → -2, Direction → -1]]
        Print[" Lim x→-2- ", Text[ fce[x]], " = ", Limit[fce[x], x → -2, Direction → 1]]
        Print["asymptota v +∞: y= " , k1 x + q1]
        Print["asymptota v -∞: y= ", k2 x + q2 ]
        der1[x_] = D[fce[x], x];
        der2[x_] = D[der1[x], x];
        Print["První derivace: y'= " , TraditionalForm[Together[der1[x]]] ]
        Print["Druhá derivace: y''= " , TraditionalForm[Together[der2[x]]] ]
        Print["Body podezřelé  z extrému: " , Solve[der1[x] == 0, x, Reals]]
        Print["Funkce je rostoucí pro x: " , TraditionalForm[Reduce[der1[x] > 0, x, Reals]]]
        Print["Funkce je klesající pro x: ", Reduce[der1[x] < 0, x, Reals] // TraditionalForm]
        Print["Body podezřelé z inflexe:", TraditionalForm[Solve[der2[x] == 0, x, Reals]]]
        Print["Funkce je konvexní pro x:", Reduce[der2[x] > 0, x, Reals] // TraditionalForm]
        Print["Funkce je konkávní pro x:", Reduce[der2[x] < 0, x, Reals] // TraditionalForm]
        Print["Průsečíky s osou x: ", TraditionalForm[Solve[fce[x] == 0, x, Reals]]]
        Print["Průsečíky s osou y: " , Rationalize[fce[0]]]
        Print[Plot[{fce[x], k1 x + q1, k2 x + q2}, {x, -10, 6}, Exclusions → {x == -2},
          ExclusionsStyle → Dashing[Small]]]
        Print[TableForm [Table[{x, fce[x]}, {x, {-4, 0}}], TableHeadings → {None, {"x", "y"}}]]
```

Obrázek 181 – Příklad 80 – grafické zadání výpisu

$$\lim_{x \to -2+} \frac{x^2}{2+x} = \infty$$

$$\lim_{x \to -2-} \frac{x^2}{2+x} = -\infty$$

asymptota v $+\infty$: $y = -2 + x$

asymptota v $-\infty$: $y = -2 + x$

První derivace: $y' = \frac{x^2 + 4x}{(x+2)^2}$

Druhá derivace: $y'' = \frac{8}{(x+2)^3}$

Body podezřelé z extrému: $\{\{x \to -4\}, \{x \to 0\}\}$

Funkce je rostoucí pro x: $x < -4 \vee x > 0$

Funkce je klesající pro x: $-4 < x < -2 \vee -2 < x < 0$

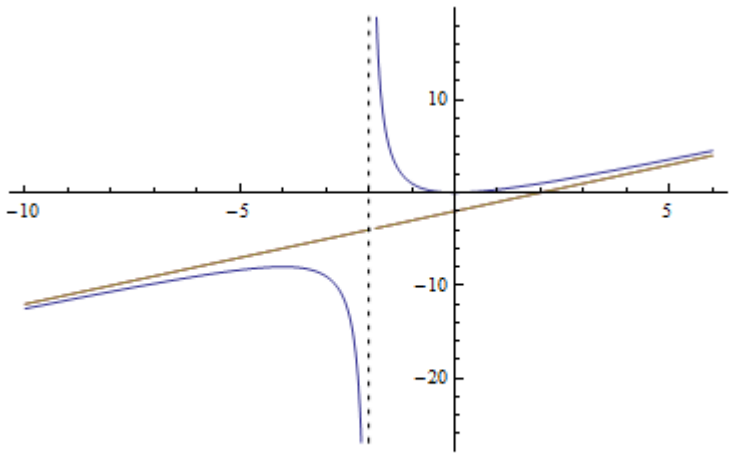
Body podezřelé z inflexe: {}

Funkce je konvexní pro x: $x > -2$

Funkce je konkávní pro x: $x < -2$

Průsečíky s osou x: $\{\{x \to 0\}, \{x \to 0\}\}$

Průsečíky s osou y: 0



x	y
-4	-8
0	0

Obrázek 182 – Příklad 80 – grafický výstup

Příklad 81

Napište rovnici tečny ke grafu funkce $f(x) = \frac{x^3 + 1}{x - 1}$ v bodě $T = [2;?]$.

V tomto příkladu potřebujeme vypočítat derivaci funkce v bodě *T*. Výsledek dosadíme do rovnice tečny $y - y_0 = f'(x_0)(x - x_0)$, kde x_0 a y_0 jsou souřadnice bodu *T*. V našem výpočtu si vytvoříme funkci *f*(*x*), obecnou derivaci *der*(*x*) a funkci tečny *tecna*(*x*) podle její definice.

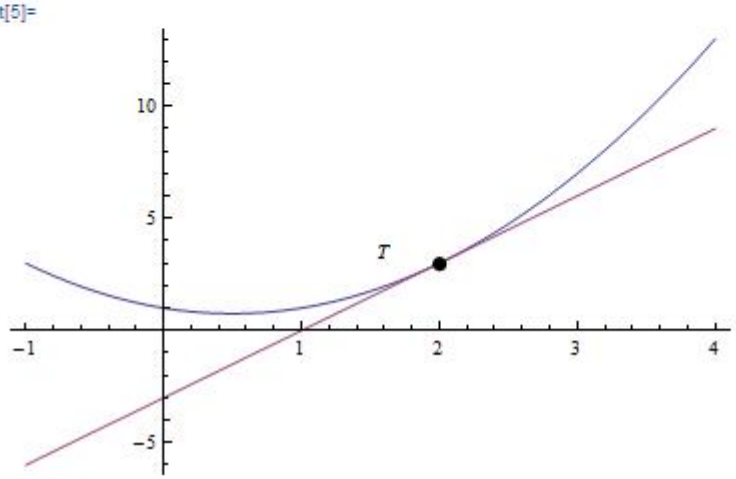
```
In[1]:= f[x_] = (1 + x^3)/(1 + x);
        der[x_] = D[f[x], x];
        tecna[x_] = der[2] (x - 2) + f[2];
        Reduce[y == tecna[x], y] // TraditionalForm
```

Out[4]//TraditionalForm=

$$y = 3x - 3$$

```
In[5]:= Plot[{f[x], tecna[x]}, {x, -1, 4},
         Epilog → {PointSize[0.02], Point[{2, f[2]}], Inset["     T"]}]
```

Out[5]=



Obrázek 183 – Příklad 81

Výsledná rovnice tečny je: $y = 3x - 3$. Direktiva Epilog v příkazu Plot[] má za úkol zobrazit bod *T*.

Příklad 82

Vypočítejte $\int \frac{27 + 36x + 13x^2}{2 + 5x + 4x^2 + x^3} \mathrm{d}x$.

Při postupech řešení tohoto integrálu se zlomek rozděluje na parciální zlomky. Jejich podobu můžeme získat příkazem Apart[].

```
In[1]:= f[x_] = (27 + 36 x + 13 x^2)/(2 + 5 x + 4 x^2 + x^3);
        Apart[f[x]] // TraditionalForm
        Integrate[f[x], x] // TraditionalForm
```

Out[2]//TraditionalForm=

$$\frac{7}{x+2} + \frac{6}{x+1} + \frac{4}{(x+1)^2}$$

Out[3]//TraditionalForm=

$$-\frac{4}{x+1} + 6\log(-x-1) + 7\log(x+2)$$

Obrázek 184 – Příklad 82

Výsledek integrálu bohužel není ideální. Nezobrazuje absolutní hodnotu v logaritmické funkci. Znaménko výrazu v logaritmu tedy můžeme změnit podle Df funkce, v tomto případě můžeme výsledek zapsat ve tvaru $6\log|x+1|+7\log|x+2|-\frac{4}{x+1}$.

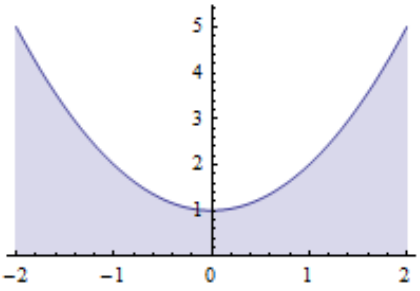
Příklad 83

Zobrazte a vypočítejte plochu vymezenou funkcí $f(x)=x^2+1$ a osou x v intervalu $x\in\langle -2,2\rangle$.

Určete objem tělesa vzniklého rotací této plochy kolem osy x a zobrazte výsledné těleso.

První část je jednoduchá, funkce Plot[] zobrazí graf funkce a Integrate[] s parametry mezí hodnoty x vypočítá velikost hledané plochy.

```
In[1]:= Plot[x^2 + 1, {x, -2, 2}, Filling → Axis, PlotRange → {0, 5.5}]
        Integrate[x^2 + 1, {x, -2, 2}]
```



Out[2]= $\frac{28}{3}$

Obrázek 185 – Příklad 83 – plocha

V druhé části se pokusíme vytvořit grafické znázornění tělesa, které vznikne rotací dané plochy. Souřadnice y se stává poloměrem v daném bodě x. Z této úvahy dostaneme pro body tělesa podmínky:

$$f(x)^2 \geq y^2+z^2,\ x\in\langle -2,2\rangle \Rightarrow y^2+z^2-f(x)^2\leq 0,\ x\in\langle -2,2\rangle.$$

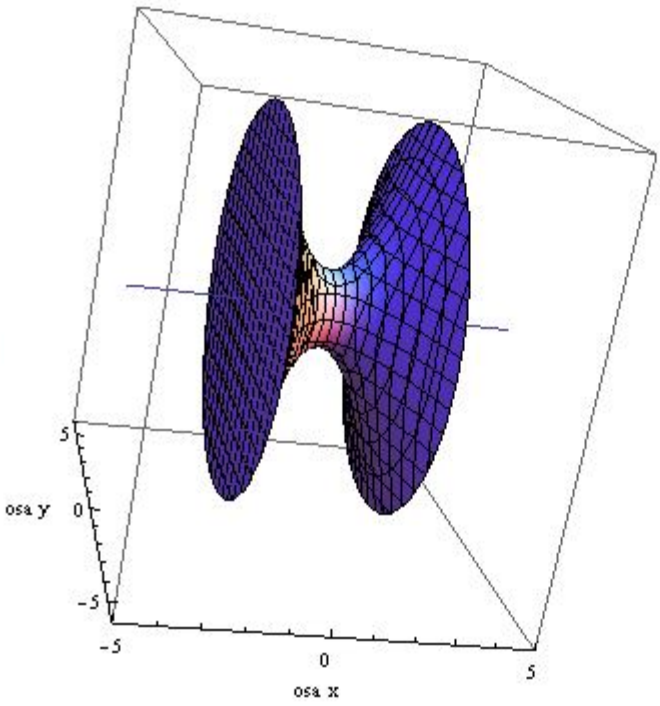
Pro objem tělesa vzniklého rotací použijeme vzorec:

$$V=\pi\int_{-2}^{2} f(x)^2\,\mathrm{d}x=\pi\int_{-2}^{2}\left(x^2+1\right)^2\,\mathrm{d}x.$$

Pro grafické znázornění jsme spojili dohromady dva grafy. První představuje vzniklé těleso a pro zobrazení jsme vybrali funkci RegionPlot3D[], která zobrazuje body splňující zadanou podmínku. Drobnou poznámkou je, že proměnné musí být na jedné straně podmínky. Druhý graf zde slouží k zobrazení osy, kolem které plochu rotujeme.

```
In[1]:= graf1 = RegionPlot3D[z^2 + y^2 - (x^2 + 1)^2 <= 0, {x, -2, 2}, {y, -6, 6}, {z, -6, 6}];
        graf2 = ParametricPlot3D[{x, 0, 0}, {x, -5, 5},
          PlotRange → {{-5, 5}, {-6, 6}, {-6, 6}}, Axes → {True, True, False},
          AxesLabel → {"osa x", "osa y", "z"}];
        Show[graf2, graf1]
        Integrate[π (x^2 + 1)^2, {x, -2, 2}]
```

Out[2]=



Out[3]= $\frac{412\,\pi}{15}$

Obrázek 186 – Příklad 83 – těleso

Výsledný objem vyšel $\frac{412}{15}\pi$ objemových jednotek.

9 Zajímavosti, co také MATHEMATICA dokáže

Program MATHEMATCA pracuje i s rozsáhlými databázemi dat. Protože tato data nepoužívá každý, využívá aplikace připojení k internetu a stažení těchto dat v případě požadavku aplikace. Přístup k internetu můžete nastavit v nabídce **Help** > **Internet Connectivity**. Výpočty v takovém případě mohou trvat déle, ale po přístupu do databáze se ostatní výpočty urychlí.

Příklad 84

Zobrazte graficky evropské státy a jejich vlajky.

```
In[1]:= CountryData["Europe"]
        Show[CountryData[#, "Flag"], ImageSize → 30] & /@ CountryData["Europe"]
        Graphics[{EdgeForm[Blue], Yellow,
          CountryData[#, "Polygon"] & /@ CountryData["Europe"]}]

Out[1]=
{Albania, Andorra, Austria, Belarus, Belgium, BosniaHerzegovina, Bulgaria,
 Croatia, Cyprus, CzechRepublic, Denmark, Estonia, FaroeIslands, Finland,
 France, Germany, Gibraltar, Greece, Guernsey, Hungary, Iceland, Ireland,
 IsleOfMan, Italy, Jersey, Kosovo, Latvia, Liechtenstein, Lithuania,
 Luxembourg, Macedonia, Malta, Moldova, Monaco, Montenegro, Netherlands,
 Norway, Poland, Portugal, Romania, SanMarino, Serbia, Slovakia, Slovenia,
 Spain, Svalbard, Sweden, Switzerland, Ukraine, UnitedKingdom, VaticanCity}
```

Out[2]=

Out[3]=

Obrázek 187 – Příklad 84 – státy Evropy

První příkaz CountryData[“Europe“] zobrazí přehled zemí v Evropě. Druhý příklad je zobrazení vlajek příkazem Show[], jednotlivé položky jsou generovány dosazováním hodnot

z prvního příkazu do položky místo znaku # ve funkci CountryData[#,“Flag“], která je zodpovědná za zobrazení vlajek států. Mapa evropských států je výsledkem podobně strukturovaného příkazu Graphics[]. Pro jednoduchost si ještě ukážeme vykreslení okolních států a zápis velikosti, populace, vlajky a velkých měst České republiky.

```
In[1]:= Graphics[{EdgeForm[Black], Yellow,
    CountryData["CzechRepublic", "Polygon"],
    EdgeForm[Black], Green, CountryData["Germany", "Polygon"],
    EdgeForm[Black], Green, CountryData["Poland", "Polygon"],
    EdgeForm[Black], Green, CountryData["Slovakia", "Polygon"],
    EdgeForm[Black], Green, CountryData["Slovakia", "Polygon"],
    EdgeForm[Black], Green, CountryData["Austria", "Polygon"]
   }]
  CountryData["CzechRepublic", "Area"]
  CountryData["CzechRepublic", "Population"]
  CountryData["CzechRepublic", "Flag"]
  CountryData["CzechRepublic", "LargestCities"]
```

Out[1]=

Out[2]=

$78\,867.$

Out[3]=

1.0411×10^{7}

Out[4]=

Out[5]=

```
{{Prague, Prague, CzechRepublic},
 {Brno, Jihomoravsky, CzechRepublic},
 {Ostrava, Moravskoslezsky, CzechRepublic},
 {Plzen, Plzensky, CzechRepublic},
 {Olomouc, Olomoucky, CzechRepublic},
 {Liberec, Liberecky, CzechRepublic},
 {CeskeBudejovice, Jihocesky, CzechRepublic},
 {HradecKralove, Kralovehradecky, CzechRepublic},
 {UstiNadLabem, Ustecky, CzechRepublic},
 {Pardubice, Pardubicky, CzechRepublic},
```

Obrázek 188 – Příklad 84 – Česká republika

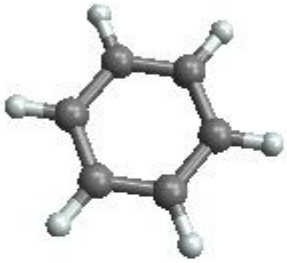
Přehled velkých měst je úplný, ale do naší prezentace se nám nevešel. Rozloha je v km^2.

Stejně jako nabízí program zeměpisná data, pracuje i s informacemi oboru chemie.

Příklad 85

Zobrazte molekulu benzenu, její vzorec, diagram a molární hmotnost.

```
In[1]:= ChemicalData["Benzene", "MoleculePlot"]
        ChemicalData["Benzene", "FormulaDisplay"]
        ChemicalData["Benzene"]
        ChemicalData["Benzene", "MolecularWeight"] " g/mol"
Out[1]=
```



```
Out[2]=
  C6H6
```

```
Out[3]=
```

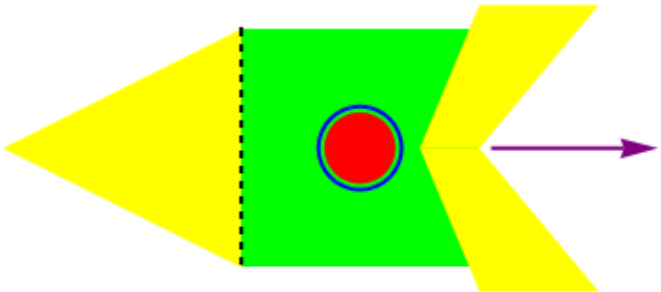
```
Out[4]=
  78.1118 g/mol
```

Obrázek 189 – Příklad 85 – benzen

Pro své aplikace můžeme vytvářet i různé grafické objekty pomocí příkazů Graphics[] a Graphics3D[].

Příklad 86

Vytvořte podle vzoru obrázek rakety.

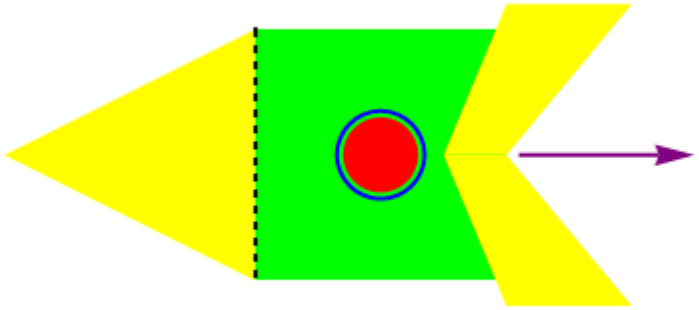


Obrázek 190 – Příklad 86 – zadání

Vytvoříme úvodní trojúhelník příkazem Polygon[] a zelený čtverec příkazem Rectangle[]. Zakreslíme modrou kružnici Circle[] a červený kruh Disk[]. Doplníme symetricky polygony a šipku Arrow[] na zadní části. Nakonec zakreslíme přerušovanou čáru Line[] na špičce rakety.

```
In[1]:= Graphics[{Yellow, Polygon[{{0, 0}, {2, 1}, {2, -1}}],
        Thick, Green, Rectangle[{2, -1}, {4, 1}], Blue,
        Circle[{3, 0}, 0.35], Red, Disk[{3, 0}, 0.3],
        Yellow, Polygon[{{3.5, 0}, {4, 0}, {5, 1.2}, {4, 1.2}}],
        Yellow, Polygon[{{3.5, 0}, {4, 0}, {5, -1.2}, {4, -1.2}}],
        Purple, Arrowheads[Large], Arrow[{{4.1, 0}, {5.5, 0}}],
        Black, Dashed, Line[{{2, 1}, {2, -1}}]
       }]
```

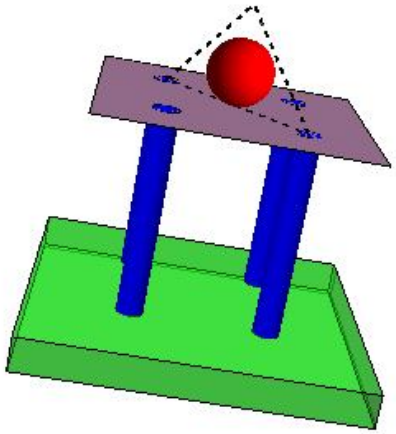
Out[1]=



Obrázek 191 – Příklad 86 – výsledek

Příklad 87

Zobrazte v 3D model stolu stojící na poloprůhledném kvádru, na kterém leží koule a čarou vytvořený trojúhelník.

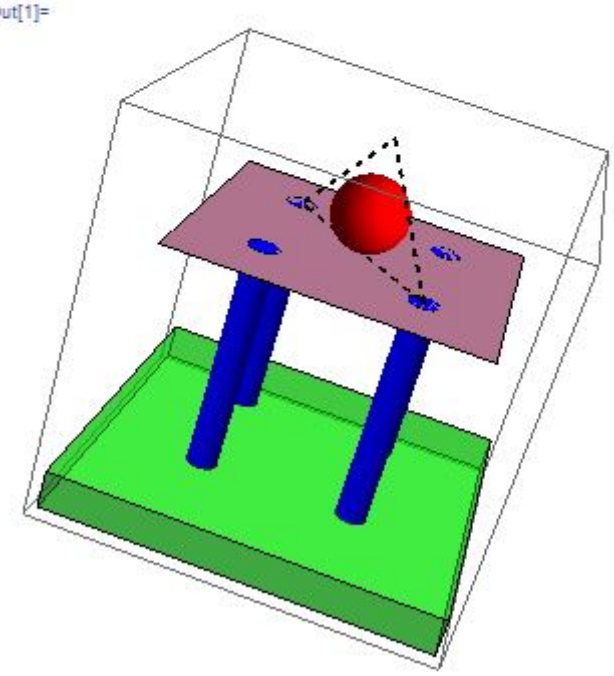


Obrázek 192 – Příklad 87 – zadání

Nohy stolu jsou válce sestrojené příkazem Cylinder[], deska stolu je polygon a podklad je kvádr Cuboid[] s průhledností Opacity[0.5]. Na stole leží koule Sphere[] a trojúhelník z lomené přerušované čáry Line[].

```
In[1]:= Graphics3D[{Blue, Cylinder[{{0, 0, 0}, {0, 0, 3}}, 0.2],
        Blue, Cylinder[{{0, 2, 0}, {0, 2, 3}}, 0.2],
        Blue, Cylinder[{{2, 0, 0}, {2, 0, 3}}, 0.2],
        Blue, Cylinder[{{2, 2, 0}, {2, 2, 3}}, 0.2],
        Red, Sphere[{1, 1, 3.5}, 0.5],
        Black, Thick, Dashed,
        Line[{{0, 0, 3}, {1, 1, 4.5}, {2, 2, 3}, {0, 0, 3}}],
        Pink, Polygon[{{-1, -1, 3}, {3, -1, 3}, {3, 3, 3}, {-1, 3, 3}}],
        Green, Opacity[.5],
        Cuboid[{-1.5, -1.5, 0}, {3.5, 3.5, -0.5}]}]
```

Out[1]=

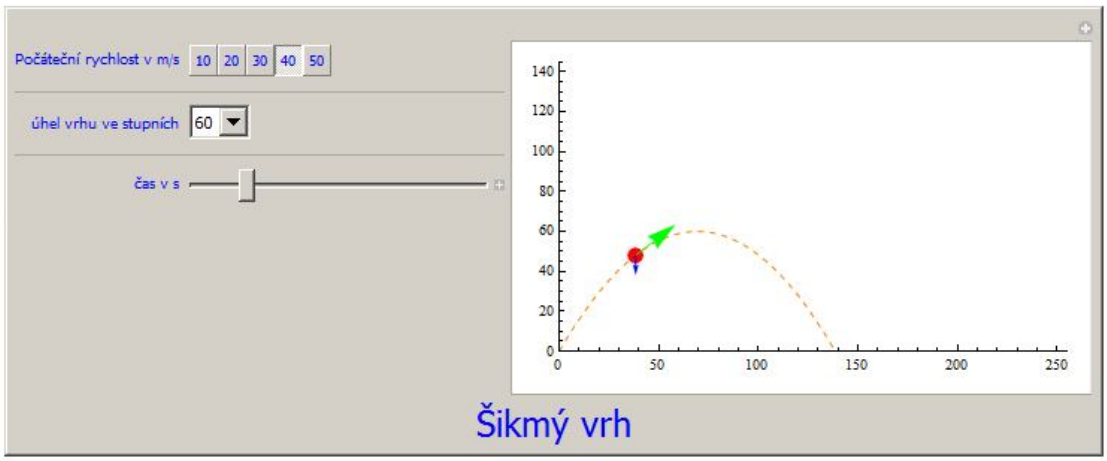


Obrázek 193 – Příklad 87 – výsledek

Grafické objekty lze vkládat do grafů a vytvořených obrázků. S použitím manipulátorů lze vytvářet dynamické modely. Na následném kódu si ukážeme vymodelování šikmého vrhu. Spojuje parametrický graf, znázornění tělesa bodovým grafem, zelenou šipku zastupující rychlost a modrou šipku přestavující zrychlení. Uživatel může nastavit počáteční rychlost, úhel vrhu a čas.

```
In[1]:= Manipulate[x[t_] = vo * Cos[α * Degree] * t;
        g = 10;
        y[t_] = vo * Sin[α * Degree] * t - 1 / 2 * g * t^2;
        vx[t_] = vo * Cos[α * Degree];
        vy[t_] = vo * Sin[α * Degree] - g * t;
        Show[ParametricPlot[{x[t], y[t]}, {t, 0, 10}, AspectRatio → Automatic, PlotStyle → Directive[Orange, Dashed],
          PlotRange → {{0, 255}, {0, 145}}],
         ListPlot[{{x[to], y[to]}}, PlotStyle → {Red, PointSize[0.03]}],
         Graphics[{Green, Arrowheads[Large], Arrow[{{x[to], y[to]}, {x[to] + vx[to], y[to] + vy[to]}}],
           Blue, Arrowheads[Small], Arrow[{{x[to], y[to]}, {x[to], y[to] - g}}]}]],
        {{vo, 40, "Počáteční rychlost v m/s"}, {10, 20, 30, 40, 50}},
        Delimiter, {{α, 60, "úhel vrhu ve stupních"}, {10, 20, 30, 40, 45, 50, 60, 70, 75, 80, 90}}, Delimiter,
        Delimiter, {{to, 0, "čas v s"}, 0, 10}, FrameLabel → Style["Šikmý vrh", Large], LabelStyle → Blue,
        ControlPlacement → Left]
```

Obrázek 194 – zdrojový kód modelu šikmého vrhu



Obrázek 195 – výsledný model šikmého vrhu

Na všech příkladech lze vidět, že program MATHEMATICA má široké využití, a to nejen v matematice a při výpočtech, ale i v ostatních předmětech. Jeho výhodou je široká podpora a spousta vytvořených aplikací k různým oborům, které lehce nalezneme na firemních stránkách demonstrations.wolfram.com.

10 Přehled vybraných příkazů

Abs[]	absolutní hodnota,
Apart[]	rozklad výrazu na parciální zlomky,
ArcCos[]	funkce $\cos^{-1}$,
ArcCot[]	funkce cotg^{-1},
ArcSin[]	funkce $\sin^{-1}$,
ArcTan[]	funkce tg^{-1},
Arg[]	argument komplexního čísla,
Arrow[]	šipka,
Arrowheads[]	hlavička šipky,
Binomial[]	kombinační číslo,
Circle[]	kružnice,
Clear[]	vymazání definice objektu,
ClearAll[]	vymaže definici vlastnosti objektu,
Conjugate[]	komplexně sdružené číslo,
ContourPlot[]	implicitně zadaný graf,
Cos[]	funkce cos,
Cot[]	funkce cotg,
CountryData[]	databáze zeměpisných dat,
Cross[]	vektorový součin,
Csc[]	funkce kosekans (csc),
Cuboid[]	kvádr,
Cylinder[]	válec,
D[]	derivace,
Dashing[]	styl přerušované čáry,
Det[]	determinant matice,
Dimensions[]	dimenze matice,
Directive[]	styl objektu,
Disk[]	kružnice,
Do[]	cyklus s řídící hodnotou,
EdgeForm[]	typ ohraničení,
Exp[]	funkce e^x,
Expand[]	roznásobení výrazu,
FindRoot[]	výpočet řešení z daného bodu iterační metodou,
Flatten[]	snížení dimenze matice,

For[]	cyklus s daným počtem opakování a řídící proměnnou,
Graphics[]	grafické zobrazení 2D objektů,
Graphics3D[]	grafické zobrazení 3D objektů,
ChemicalData[]	databáze chemických látek,
Im[]	imaginární část komplexního čísla,
Integrate[]	výpočet integrálu,
KSubsets[]	vytvoření podmnožin ze seznamu o daném počtu prvků,
Length[]	počet prvků seznamu (pole),
Limit[]	výpočet limity,
Line[]	lomená čára,
ListPlot[]	vykreslení grafu pomocí zadaných bodů,
Log[]	funkce log,
Manipulate[]	nástroj na nastavení vstupních parametrů,
N[]	numerická hodnota výrazu s udanou přesností,
Norm[]	velikost vektoru,
Normalize[]	normovaný (jednotkový) vektor ve směru zadaného vektoru,
NSolve[]	numerický výpočet,
Opacity[]	průhlednost objektu,
Outer[]	vytvoření uspořádaných *n*-tic,
ParametricPlot[]	vykreslení parametricky zadaného grafu,
ParametricPlot3D[]	parametrický graf v 3D,
Permutations[]	permutace prvků dané třídy,
Plot[]	vykreslení klasického grafu,
Plot3D[]	vykreslení prostorového grafu,
Point[]	bod zadaný souřadnicemi,
PointSize[]	velikost bodu,
PolarPlot[]	polárně zadaný graf,
Polygon[]	polygon zadaný vrcholy,
Power[]	zadaná mocnina zadaného výrazu,
Print[]	zápis výrazu,
Random[]	náhodná čísla,
Re[]	reálná část komplexního čísla,
Rectangle[]	obdélník,
Reduce[]	řešení rovnic a nerovnic,
RegionPlot3D[]	vykreslení oblasti platnosti podmínky v 3D grafu,

RGBColor[]	barva zadaná v RGB,
Sec[]	funkce sekans (sec),
Select[]	výběr ze seznamu podle kritéria,
Show[]	zobrazí sadu grafických objektů,
Sign[]	funkce sign, v **C** komplexní jednotka s argumentem výrazu,
Simplify[]	zjednodušení výrazu,
Sin[]	funkce sin,
Solve[]	řešení rovnic,
Sort[]	seřazení,
Sphere[]	koule,
Sqrt[]	odmocnina z výrazu,
StreamPlot[]	typ vektorového grafu,
Sum[]	součet výrazů,
Table[]	vytvoření tabulky,
TableForm[]	režim zobrazení tabulky,
Tan[]	funkce tg,
Together[]	sloučení výrazu na společného jmenovatele,
TraditionalForm[]	zobrazení výrazu v tradičním matematickém zápisu,
VectorAngle[]	úhel mezi vektory,
VectorPlot[]	vektorový graf,
VectorPlot3D[]	vektorový graf v 3D,
While[]	cyklus s podmínkou na začátku.

11 Přehled menu

Přehled vybraných položek hlavního menu.

File:

New (Ctrl+N) – vytvoření nového notebooku;

Open (Ctrl+O) – otevření předtím vytvořeného souboru;

Close (Ctrl+F4) – zavření aktuálního notebooku;

Close All (Shift+Ctrl+F4) – zavření všech notebooků;

Save (Ctrl+S) – uložit aktuální notebook;

Save As (Shift+Ctrl+S) – uložit notebook pod jiným názvem;

Save Selection As – uložení výběru;

Revert – vrátit se k předchozí verzi souboru;

Install – instalace doplňků;

Send To – odeslání výběru;

Printing Settings – nastavení tisku;

Print – tisk;

Exit – ukončení aplikace.

Edit:

Undo (Ctrl+Z) – krok zpět;

Cut (Ctrl+X),

Copy (Ctrl+C),

Paste (Ctrl+V),

Clear (Delete),

Copy As – vyjmout, kopírovat, vložit, odstranit, …;

Extend Selection – rozšířit výběr;

Select All – vyber vše;

Check Balance – kontrola zbytku;

Un/Comment Selection – nastavení výběru komentáře;

Complete Selection (Ctrl+K) – kompletní výběr z rezervovaných slov;

Make Template (Shift+Ctrl+K) – vytvoření šablony;

Check Spelling (Alt+;) – kontrola pravopisu;

Find (Ctrl+F),

Find Next (F3),

Find Previos (Shift+F3) – vyhledávání;

Preferences – nastavení prostředí.

Insert:

Input from Above – vložení předchozího vstupu;

Output from Above – vložení předchozího výstupu;

Cell with Same Style – vložení stejné buňky;

Special Character – panel speciálních znaků;

Color – vložení barvy z dialogu;

Citation – vložení citace;

Typesetting – sazba matematických formátů;

Table/Matrix – tabulka, matice;

Horizontal Lines – oddělení čarou;

File Path – pracovní adresář;

Picture – obrázek;

File – soubor;

Object – objekt;

Hyperlink – hypertextové odkazy;

Automatic Numbering – automatické číslování;

Page Break – konec stránky.

Format:

Style – výběr stylu podle typu buňky;

Clear Formatting – zrušení formátu;

StyleSheet,

Screen Environment,

Edit Stylesheet – sešit stylů, nastavení zobrazení, nastavení stylů;

Font, Face, Size – font písma, řez písma, velikost písma;

Text Color – barva písma;

Background Color – barva pozadí;

Text Alignment,

Text Justification – zarovnání textu;

Word Wrapping – zalomení řádků.

Cell:

Convert To – upravit typ buňky;

Cell Properties – vlastnosti buněk;

Cell Tags – označení buněk;

Grouping – seskupování buněk;

Divide Cell – rozdělení buňky;

Merge Cells – sloučení buněk;

Notebook History – historie činnosti;

Delete All Output – smazat výpočty;

Show Expression – zobrazení vzorce zápisu.

Graphics:

New Graphic – nový grafický prvek;

Drawing Tools – panel nástrojů na kreslení;

Alignment Guides Enabled – povolení průvodce zarovnání;

Rendering – rendrování;

Group – seskupení objektů;

Ungroup – zrušení seskupení;

Move to Front – posunout do popředí;

Move to Back – posunout do pozadí;

Move to Forward – přesunout dopředu;

Move to Backward – přesunout dozadu;

Align Left Sides,

Align Center Vertically, … – zarovnání podle jednotlivých směrů;

Distribute … – rozmístění v jednotlivých směrech.

Evaluation:

Evaluate cells – výpočet buněk (Shift+Enter);

Evaluate in Place – výpočet daného místa (Shift+Ctrl+Enter);

Evaluate in Subsession – výpočet v dílčích prostorech;

Evaluate Notebook – výpočet sešitu;

Evaluate Initialization Cells – výpočet inicializační buňky;

Dynamic Updating Enabled – povolení dynamického přepočítávání buněk;

Convert Dynamic to Literal – doslovná dynamická konverze;

Debugger – ladění, označení Debug v popisku buněk;

Debugger Controls – ladění podle příkazů;

Interrupt Evaluation – zastavení výpočtu;

Abort Evaluation – ukončení výpočtu;

Remove from Evaluation Quene – odstranění z výpočtu;

Find Currently Evaluating Cell – vyhledání aktuálně vypočítávané buňky;

Kernel Konfiguration Options – nastavení výpočtového jádra;

Parallel Kernel Configuration,

Parallel Kernel Status – konfigurace a status paralelního jádra;

Default Kernel – standardní výpočtové jádro;

Notebook's Kernel,

Notebook's Default Contex – nastavení výpočtového jádra notebooku;

Start Kernel,

Quit Kernel – start a ukončení výpočtového jádra.

Palettes:

Basic Math Assistant,

Clasroom Assistant,

Writing Assistant – základní palety matematických a textových funkcí;

Slide Show – nastavení animace;

Chart Element Schemes – nastavení grafů;

Color Schemes – paleta barev;

Special Characters – speciální znaky;

Other – staré panely nástrojů;

Stylesheets – nastavení stylu notebooku;

Generate Palette from Selection – vytvoření palety z výběru;

Generate Notebook from Palette – vytvoření notebooku z palet.

Windows:

Magnification – zvětšení;

Show Rules – pravítko;

Show Toolbar – paleta nástrojů;

Show Handwriting Input Wide – ruční zadávání výrazů;

Stack Windows – okna za sebou;

Tile Windows Wide – okna nad sebou;

Tile Windows Tall – okna vedle sebe;

Full Screen – celá obrazovka (F12);

další nabídky představují otevřená okna.

Help:

Documentation Center (F1) – manuál;

Function Navigator – přehled funkcí;

Virtual Book – virtuální příručka;

Find Selected Function – rychlá nápověda k výběru;

Wolfram Website – web firmy Wolfram;

Demonstrations – ukázky hotových příkladů;

Internet Conectivity – nastavení připojení internetu, přístup k externím datům;

Give Feedback – kontakt;

Register this Mathematica – registrace;

Welcome Screen – úvodní obrazovka.

12 Použitá literatura

DOBRAKOVOVÁ, Jana, Monika KOVÁČOVÁ a Viera ZÁHONOVÁ. *Mathematica 5.2 pre stredoškolských učiteľov*: *študijné materiály*. 1. Bratislava: Slovenská technická univerzita v Bratislavě, 2006. ISBN 80-967305-2-5.

DOBRAKOVOVÁ, Jana, Monika KOVÁČOVÁ a Viera ZÁHONOVÁ. *Mathematica 5.2 pre stredoškolských učiteľov*: *tréningové materiály*. 1. Bratislava: Slovenská technická univerzita v Bratislavě, 2008. ISBN 80-969562-2-1.

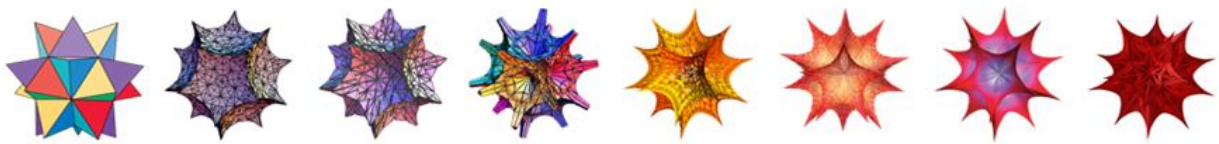
Matematické programy a jejich použití: Mathematica 7 – Základy používání programu. BC. KUSÁK, Radim. *Http://utf.mff.cuni.cz*: *Matematické programy* [online]. Praha, 2010 [cit. 2012-02-05]. Dostupné z: http://utf.mff.cuni.cz/~kusak/math.php?zalozka=1

MATHEMATICA: Mathematica – příklady. ELKAN, spol. s r.o. *Wolfram Mathematica* [online]. Praha, 2011 [cit. 2012-02-05]. Dostupné z: http://www.mathematica.cz/dokumenty-tutorial.php

REICHL, Jaroslav a Martin VŠETIČKA. Mathematica fórum. *Mathematica fórum* [online]. Praha, 2008, 29. ledna 2010 19:22:18 [cit. 2012-02-05]. Dostupné z: http://www.mathematica-forum.cz/

WOLFRAM MATHEMATICA. *Wolfram Mathematica*: *Wolfram Mathematica 8 Documentation* [online]. 30. října 2011 [cit. 2012-02-05]. Dostupné z: http://reference.wolfram.com/mathematica/guide/Mathematica.html

SLAVÍK, Antonín. Informace pro studenty: Mathematica pro začátečníky. *Http://www.karlin.mff.cuni.cz* [online]. Praha, 27. ledna 2012 [cit. 2012-02-05]. Dostupné z: http://www.karlin.mff.cuni.cz/~slavik/info.html



Logo MATHEMATICA Dostupné z: http://www.karlin.mff.cuni.cz/~slavik/info.html