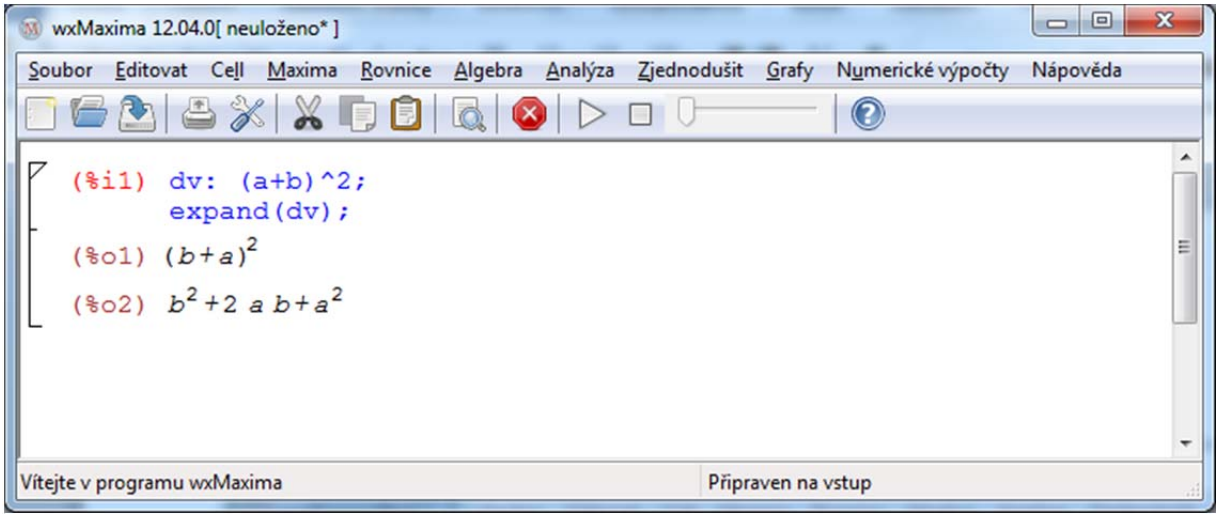
Tři příklady v programu wxMaxima

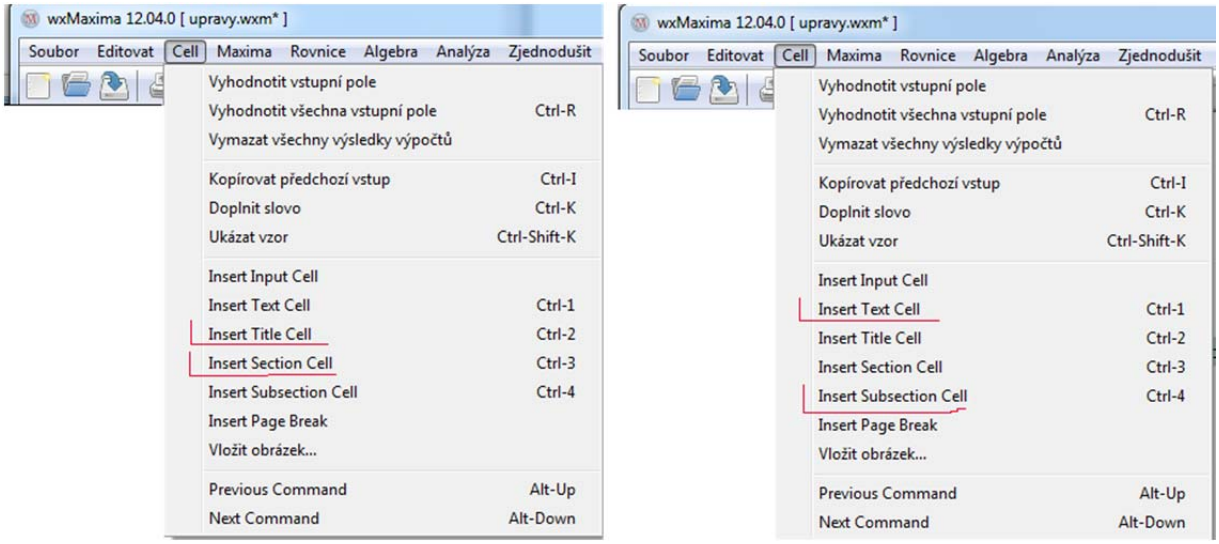
Příklad prvý – úpravy výrazu

Prvý příklad budeme bohatě ilustrovat obrázky. Doprovodíme jimi každý krok v pracovním listu.



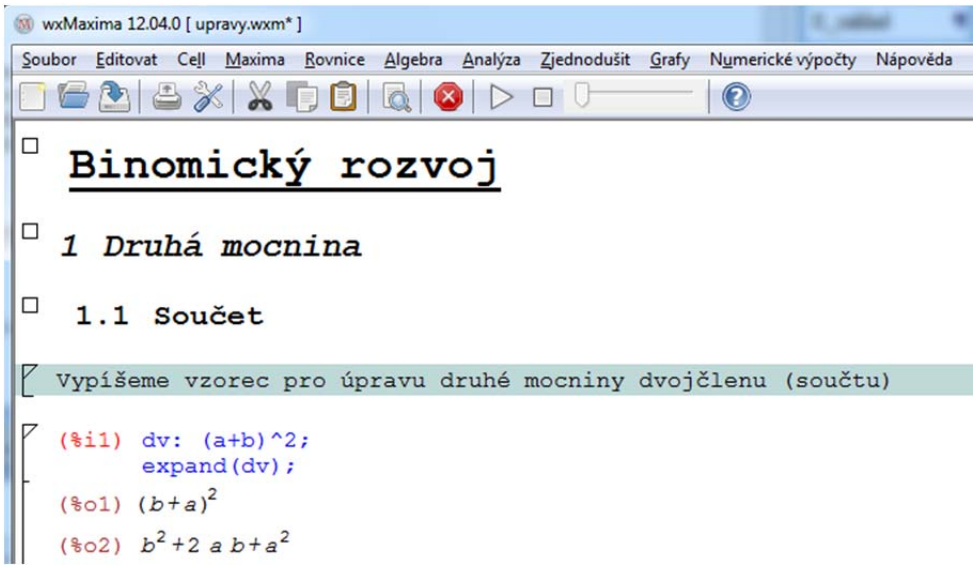
Obr. 1.1

1. Spustíme program wxMaxima. Budeme procvičovat úpravu výrazu podle vzorce – druhou mocninu dvojčlenu. Do volné plochy proto zapíšeme:
   dv: (a+b)^2; tímto příkazem vytvoříme proměnnou *dv* a uložíme do ní požadovaný výraz
   expand(dv);. výraz potom roznásobíme.
   Výsledek vstupu i jeho vyhodnocení vidíme na obrázku 1.1.
2. Takový příklad ale nebude příliš názorný. Doplníme tedy do něj doprovodné texty a rozčleníme ho na kapitoly. Myší klikneme nad pole vymezující prvý vstup (ano, je to možné) a z menu *Cell* vyvoláme položky pro vložení polí, které rozčlení dokument na části, kapitoly a podkapitoly, a které umožní vložit doprovodný text (obr. 1.2). Kam v dokumentu klikneme myší, tam se objeví vodorovná čára (autoři programu ji nazývají *vodorovný kurzor*) a tam se vloží vybrané pole.



Obr. 1.2

3. Vložíme do dokumentu hlavní titulek *Binomický rozvoj*, titulek prvé *kapitoly Druhá mocnina* a její prvé části *Součet*. O číslování se program stará sám. Pod titulek podkapitoly vložíme prvý vysvětlující text. Výsledek ukazuje obrázek 1.3. Číslování vstupů a výstupů se při uvedených úpravách (vkládání textu) pochopitelně nemění.

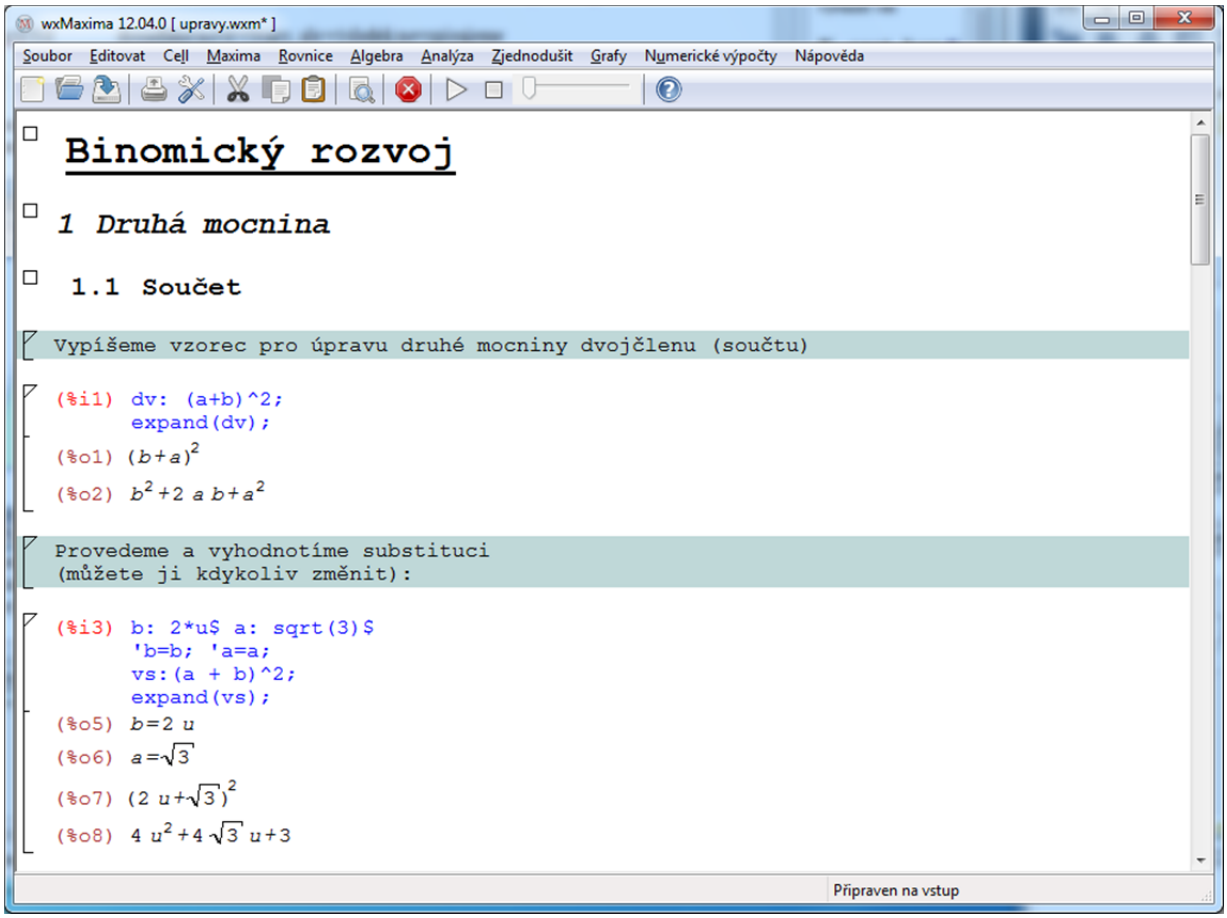


Obr. 1.3

4. Nyní do vzorce dosadíme za proměnné *a*, *b* složitější výrazy a druhou mocninu dvojčlenu upravíme podle vzorce:

*b: 2*u$ a: sqrt(3)$*	dosadíme nové výrazy, ale výstup nevypisujeme (díky znaku $)
'b=b; 'a=a;	vypíšeme substituci (viz obrázek 1.4)
vs:(a + b)^2;	zadáme a vyhodnotíme dosazený výraz (zobrazí se výstup)
expand(vs);	a roznásobíme jej (zobrazí se výstup).

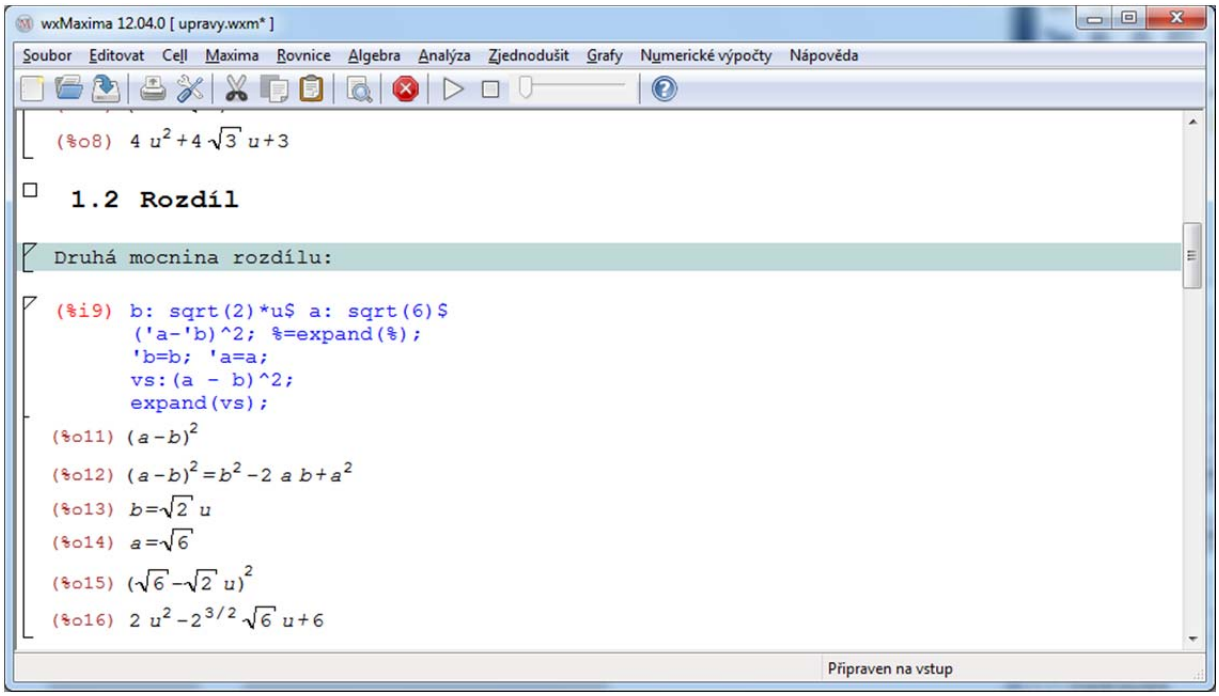
Celou skupinu výrazů vyhodnotíme (***Ctrl* + *Enter***) najednou.
Dosazení jsme také uvedli textovým komentářem.



Obr. 1.4

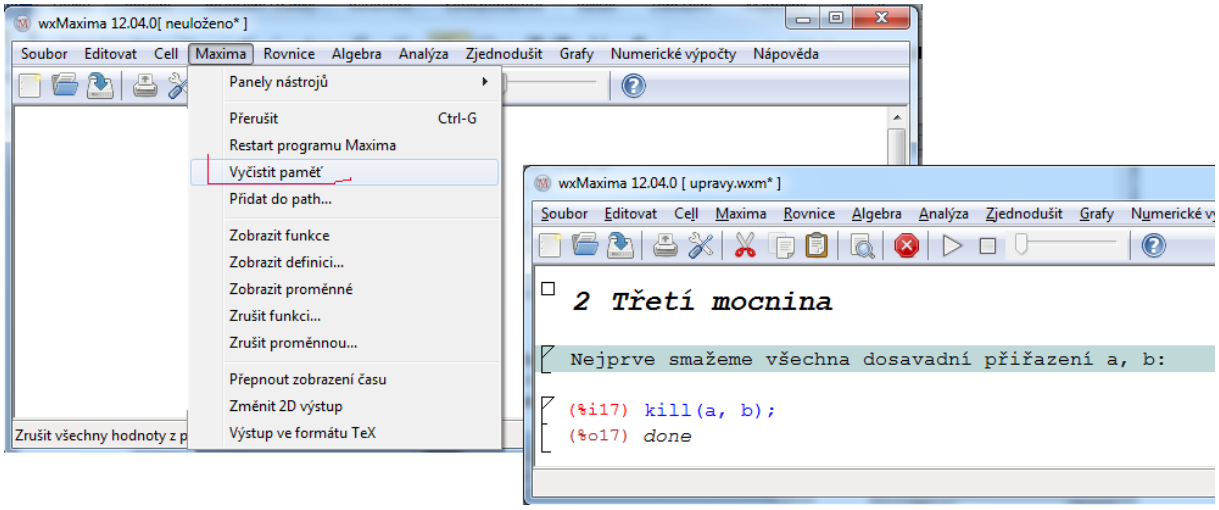
5. Podobně připravíme druhou mocninu rozdílu (obrázek 1.5):

*b: sqrt(2)*u$ a: sqrt(6)$*	dosazení za *a, b* (bez výpisu)
('a–'b)^2; %=expand(%);	výpis výrazu a jeho roznásobení
'b=b; 'a=a;	výpis substitucí
vs:(a – b)^2;	umocňovaný výraz
expand(vs);	a jeho roznásobený tvar



Obr. 1.5

6. V další kapitole budeme zkoumat vzorec $(a + b)^3$. Chceme začít úplně od začátku, potřebujeme proto zrušit všechna případná přiřazení použitých proměnných (názvů). To můžeme udělat buď rovnou z menu *Maxima* příkazem *Vyčistit paměť*, nebo, citlivěji, příkazem, v němž určíme, které proměnné se mají „uvolnit“, „zapomenout“ (obr. 1.6).

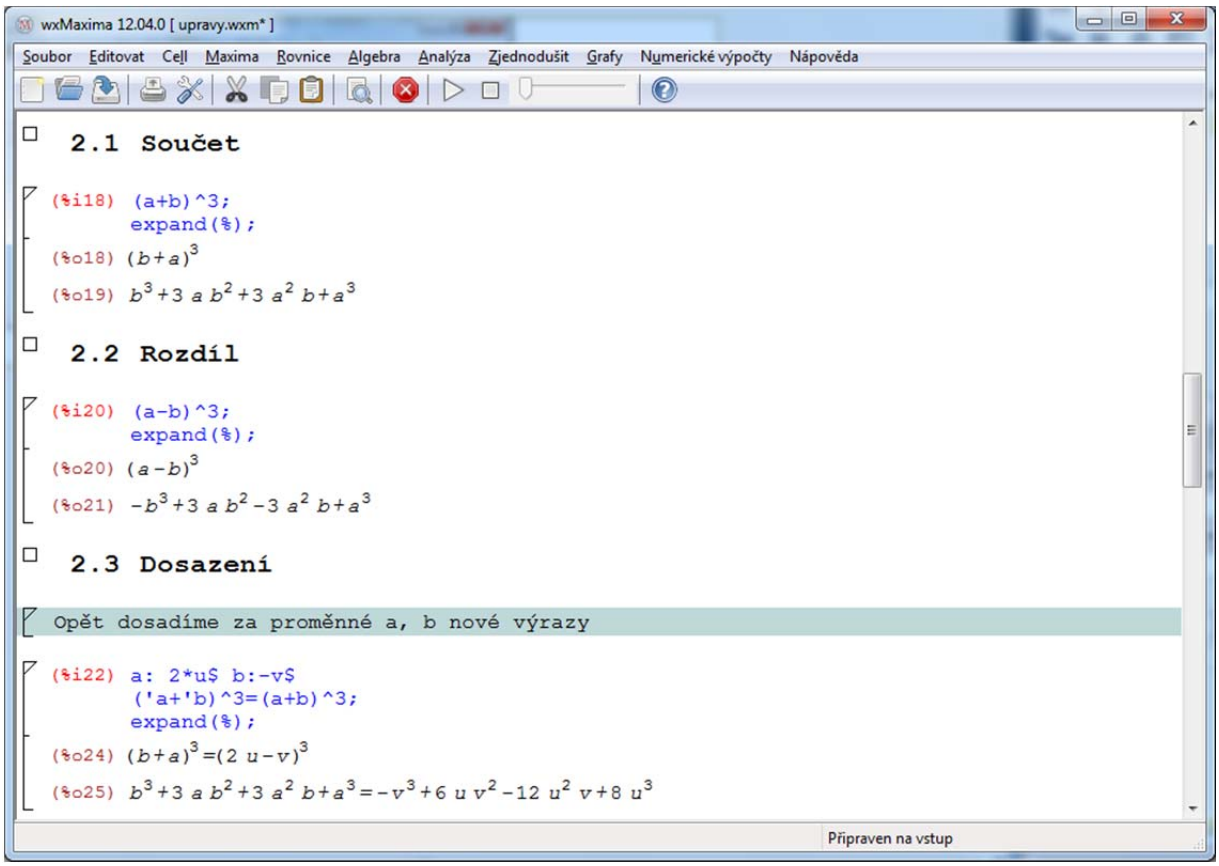


Obr. 1.6

7. Připravíme vzorce pro třetí mocninu dvojčlenu, nejprve oba pro součet i rozdíl. Zjištění, že jde o týž vzorec, je pro mnohé žáky v začátku obtížně pochopitelné.
   Poté do vzorce dosadíme a ukážeme souvislost obou tvarů (obrázek 1.7).
   Do dokumentu doplníme texty a členění do podkapitol.
   Všimněme si zápisu rovnosti ve vstupu *(%i22)*:

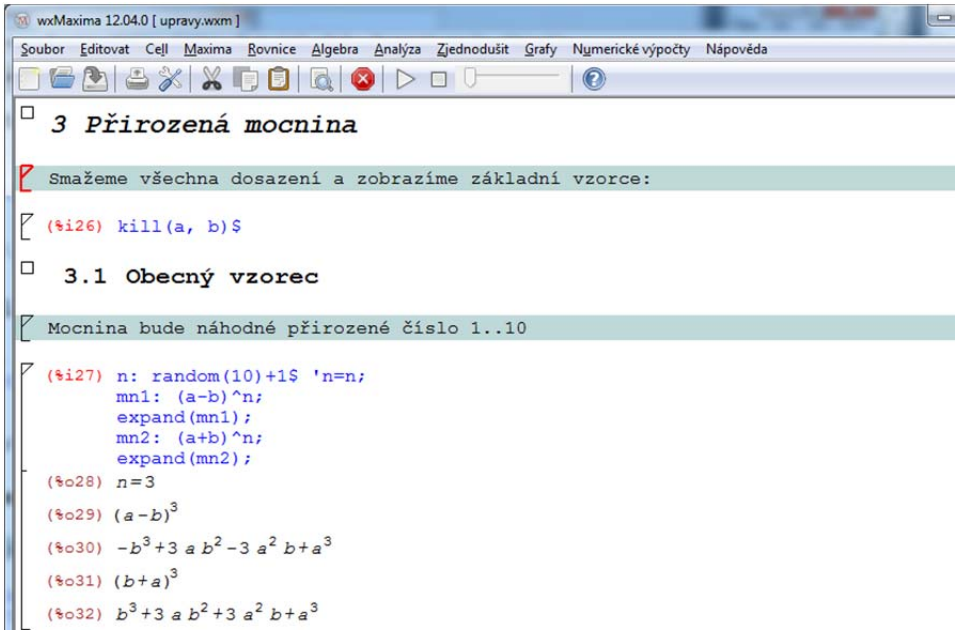
('a+'b)^3=(a+b)^3; zapíše vedle sebe jak nedosazený výraz (s názvy a, b), tak výraz po provedené substituci (a mezi výrazy zapíše rovnost)

expand(%); a vyhodnotí výše popsaný, naposledy vyhodnocený, výraz



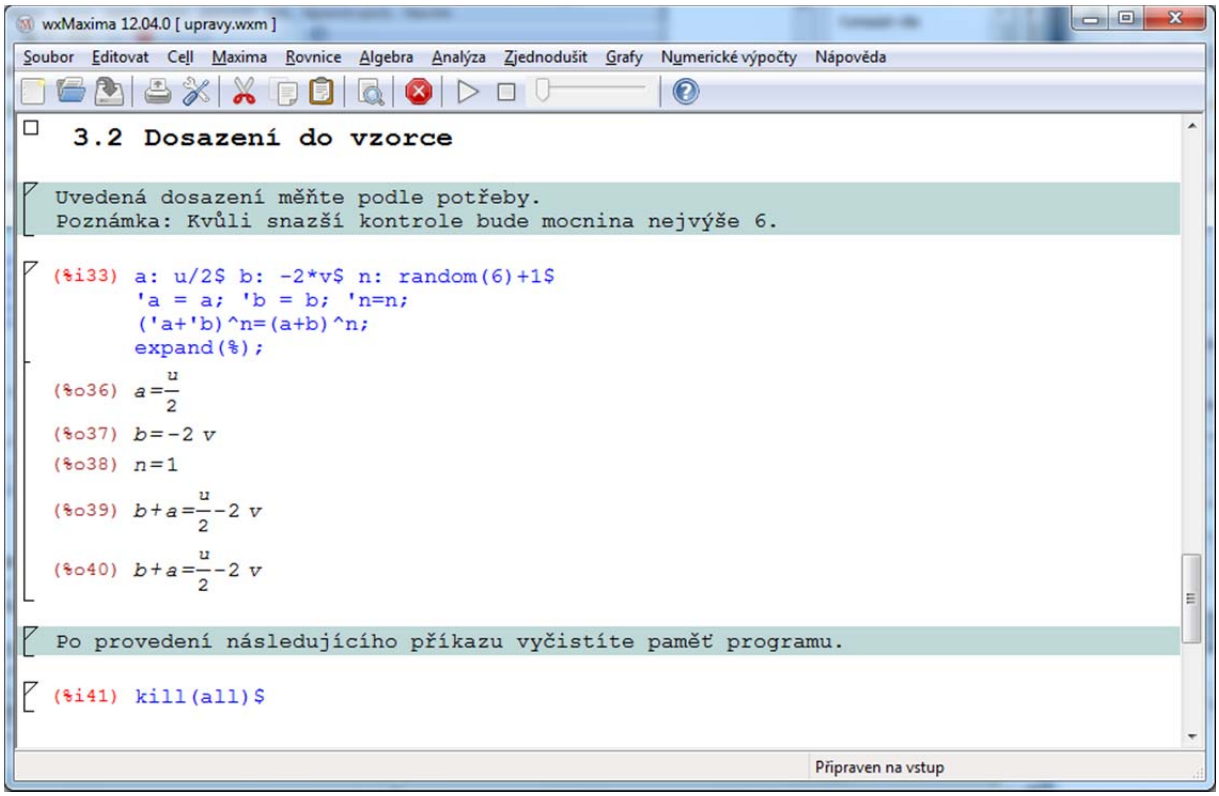
Obr. 1.7

8. V poslední kapitole se věnujeme umocnění dvojčlenu na přirozený exponent. Tento exponent nebude konstantní, ale bude se při vyhodnocení měnit – je dán náhodným číslem, které generujeme příkazem *n: random(10)+1$*, a je vypsán příkazem *'n=n;*.
   Napřed ale opět uvolníme proměnné a, b a připravíme strukturu dokumentu (obr. 1.8).



Obr. 1.8

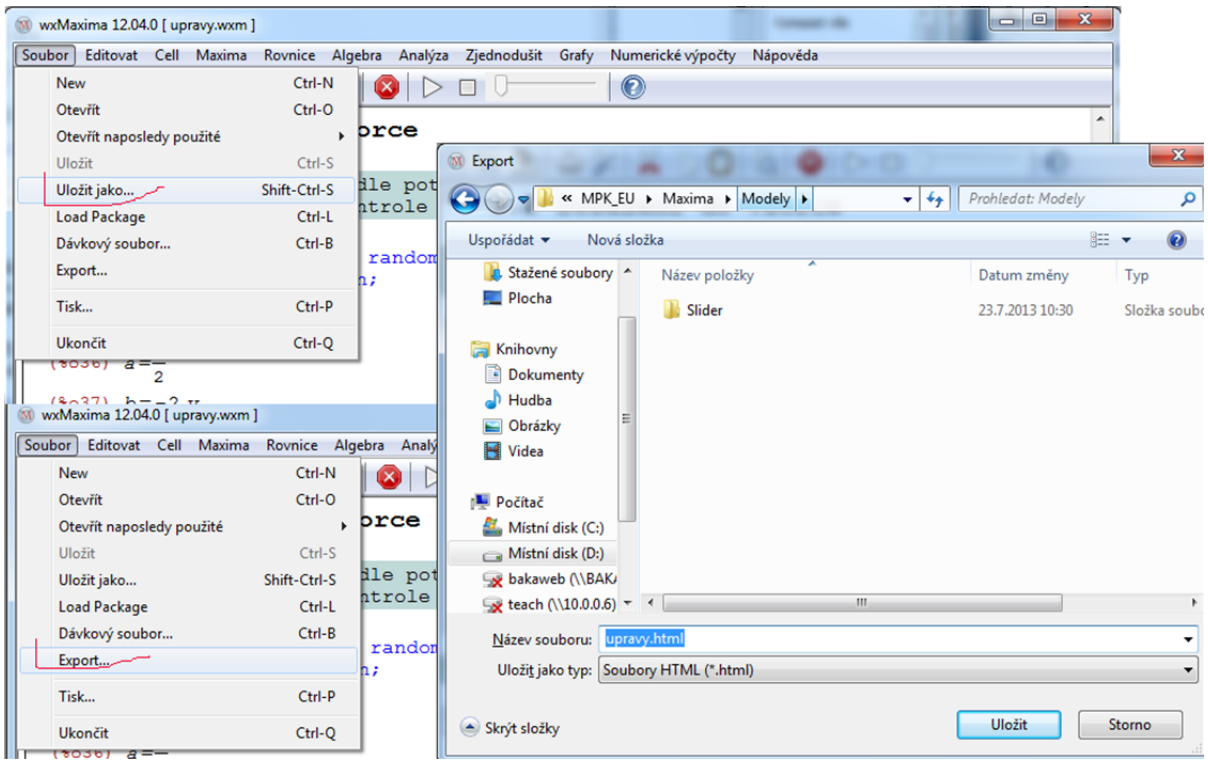
9. Za proměnné a, b opět dosadíme složitější výrazy (obrázek 1.9).



Obr. 1.9

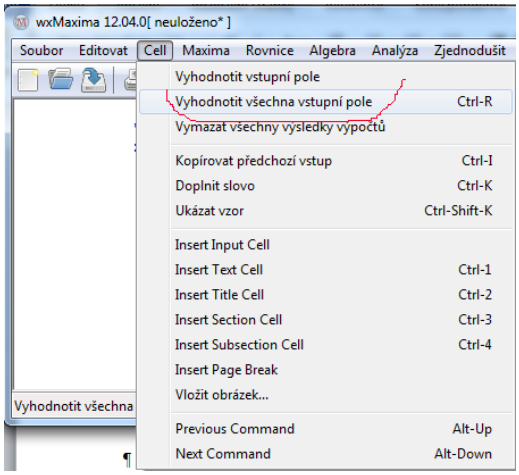
10. Na závěr můžeme přidat příkaz, kterým zcela uvolníme paměť. To má význam tehdy, jestliže budeme celý dokument opakovaně přepočítávat (bezpečnější by bylo mít příkaz *kill(all)* na úplném začátku dokumentu).

Uložení pracovního listu

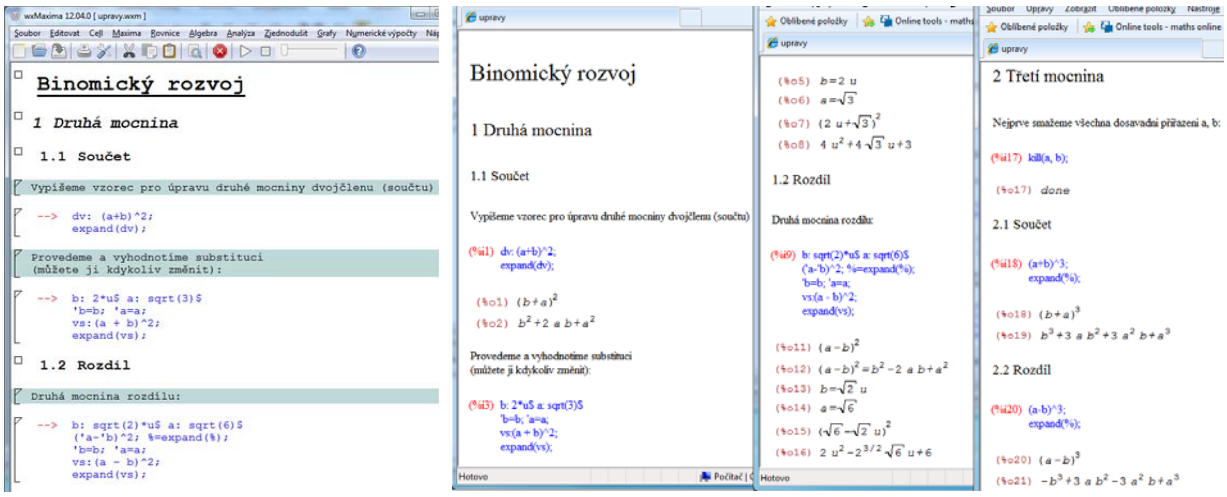


Obr. 1.10

Pracovní list je hotov. Můžeme ho uložit ve formátu wxMaxima (soubor s příponou wxm). Můžeme ho také exportovat jako webovou stránku (soubor s příponou html) – viz obrázek 1.10.
Model wxm se uloží bez vyhodnocených výsledků, bez výstupních polí (obrázek 1.12). Vyhodnocení všech příkazů provedeme příkazem *Vyhodnotit všechna výstupní pole* (zkratka ***Ctrl + R***) z podmenu *Cell* (viz náhled na obrázku 1.11). Tímto příkazem můžeme nechávat opakovaně přepočítat celý list v příslušně návaznosti vstupních a výstupních polí. (V tomtéž menu najdeme také příkaz *Vymazat všechny výsledky výpočtů*.)
Html stránka (list) se uloží ve výsledném tvaru, tedy včetně výstupních polí (kousek náhledu viz kombinovaný obrázek 1.13).



Obr. 1.11



Obr. 1.12, 1.13

Sestrojený pracovní list a příslušný html dokument najdete v souborech *upravy.vwm* a *upravy.html*. Prvý z nich otevřete – a můžete s ním libovolně pracovat – v programu wxMaxima, druhý jmenovaný soubor otevřete ve webovém prohlížeči.

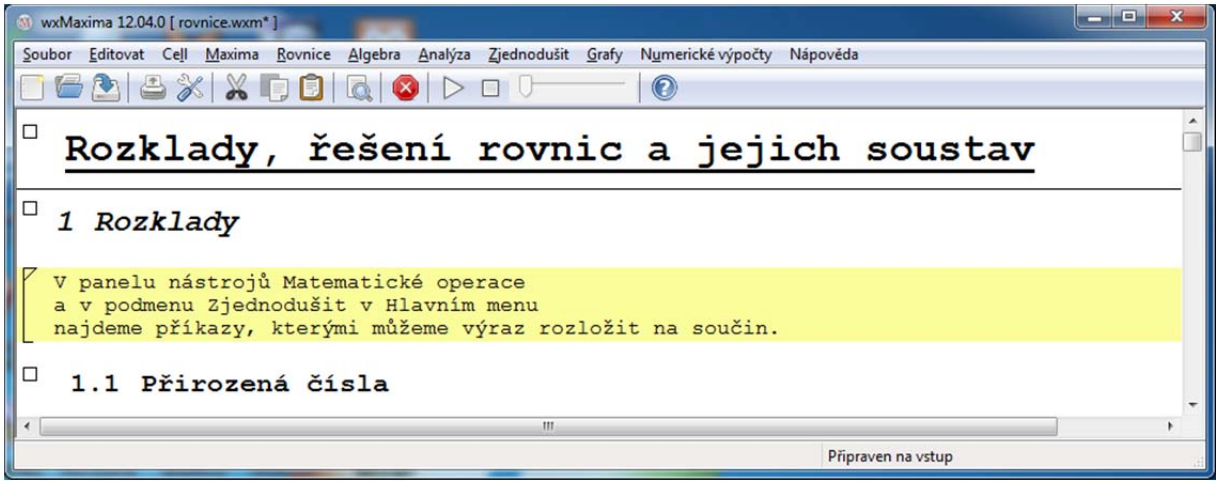
Příklad druhý – rozklady, rovnice a jejich soustavy

Druhý příklad budeme také doplňovat doprovodnými obrázky a zdůrazníme další postupy a nástroje, o nichž jsme se dosud nezmínil1.

Rozklady

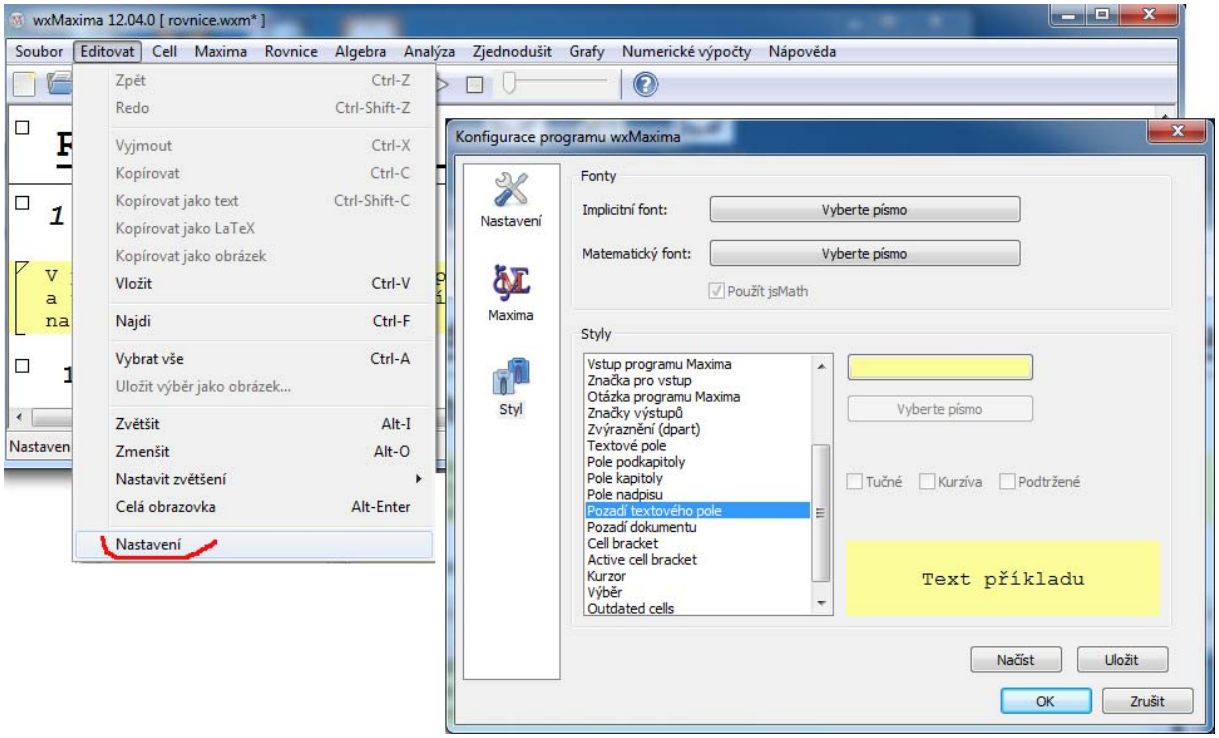
V minulém příkladu jsme v programu wxMaxima výrazy roznásobovali. Teď budeme čísla a mnohočleny naopak převádět na součinový tvar.

1. Nejprve si opět připravíme strukturu dokumentu (obr. 2.1).



Obr. 2.1

2. Pokud se nám nelíbí nastavení prostředí pracovního listu, můžeme je v podmenu *Editovat* v *Hlavnim menu* v položce *Nastavení* upravit podle vlastních preferencí. My jsme změnili barvu pozadí textových polí (obr. 2.2)

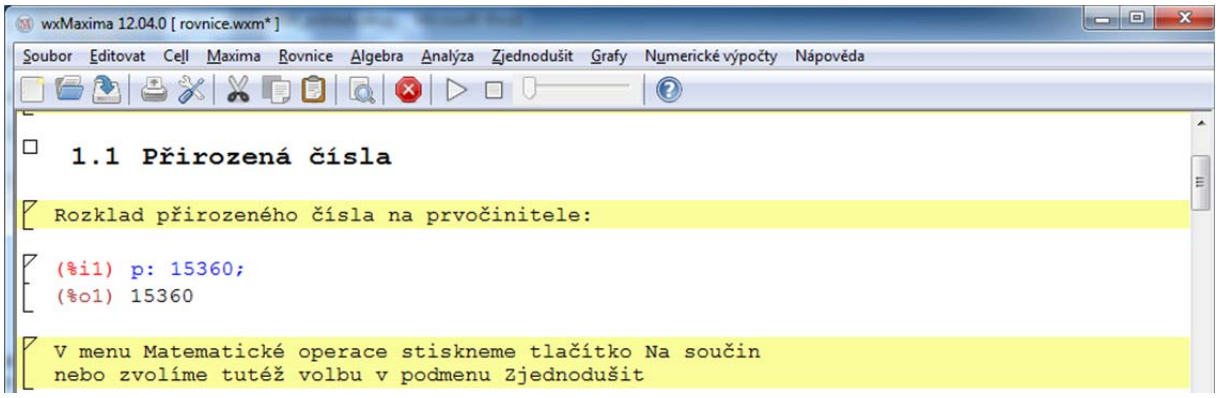


Obr 2.2

Přirozená čísla

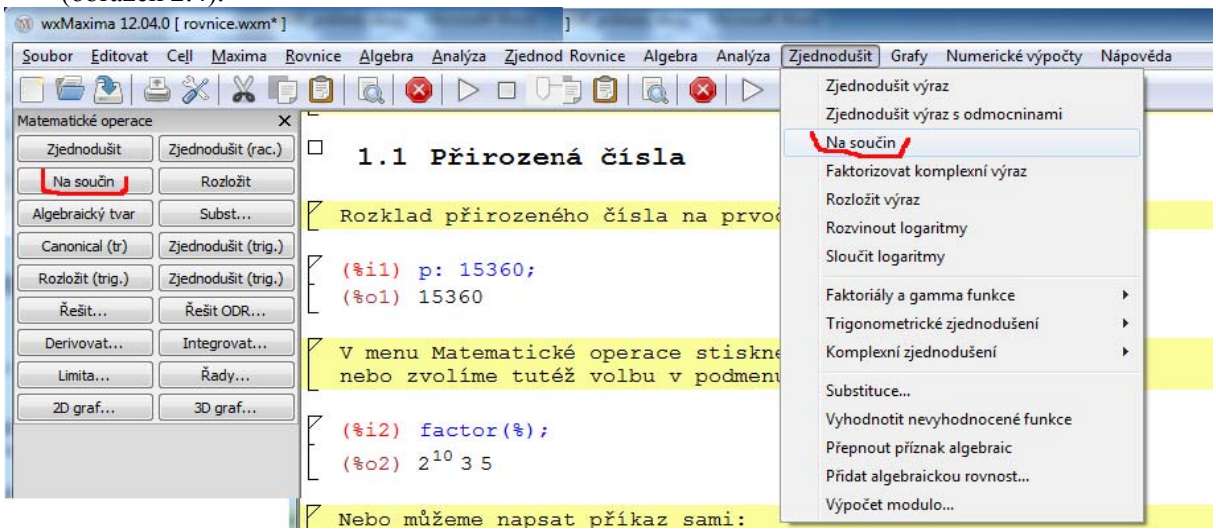
Pro práci s přirozenými čísly je v Maximě k dispozici velké množství funkcí a příkazů. Začneme s rozklady a necháme si zvolené číslo rozložit na prvočinitele. Podobně jako v prvém příkladu zkusíme nejprve využít prostředí wxMaxima a funkci vložíme pomocí *Panelu příkazů Matematické operace* či pomocí podmenu *Zjednodušit* z *Hlavního menu*.

3. Do vstupního pole zapíšeme číslo. V ukázce jsme vložili číslo 15360. Zápis jsme ukončili středníkem, vyhodnocený výstup bude tedy zobrazen (obr. 2.3) jako výstup %o1.



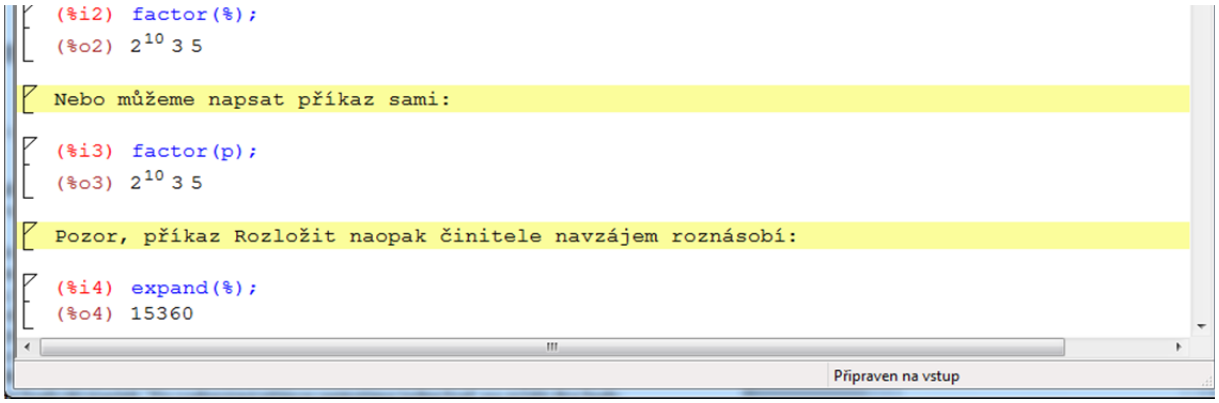
Obr. 2.3

4. Na tento výstup uplatníme příkaz (funkci) *Na součin*. Najdeme ji jak v Panelu, tak v menu. (obrázek 2.4).



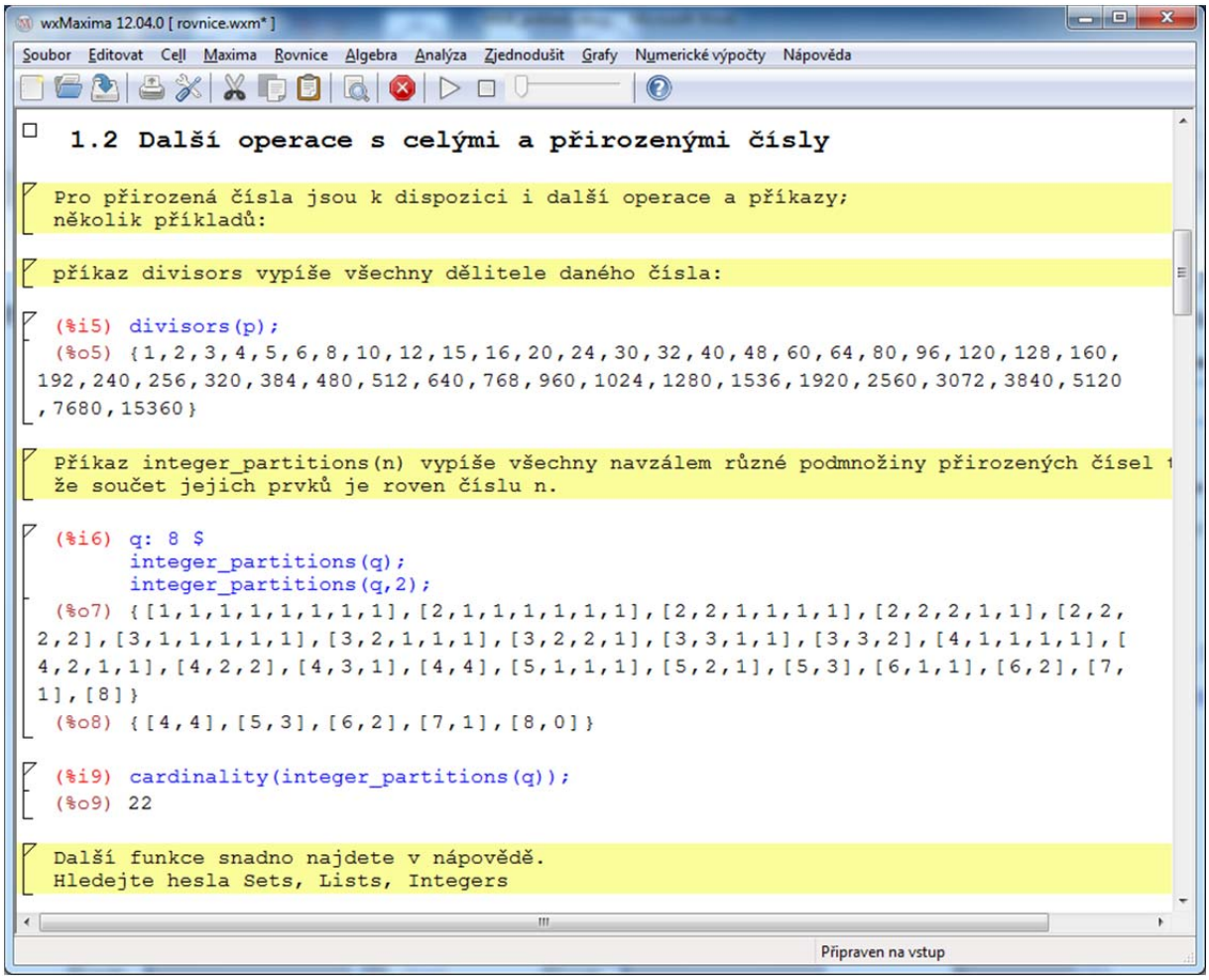
Obr. 2.4

5. Zápis příkazu vidíme v pracovní ploše programu, příště můžeme psát příkaz rovnou (obr. 2.5).

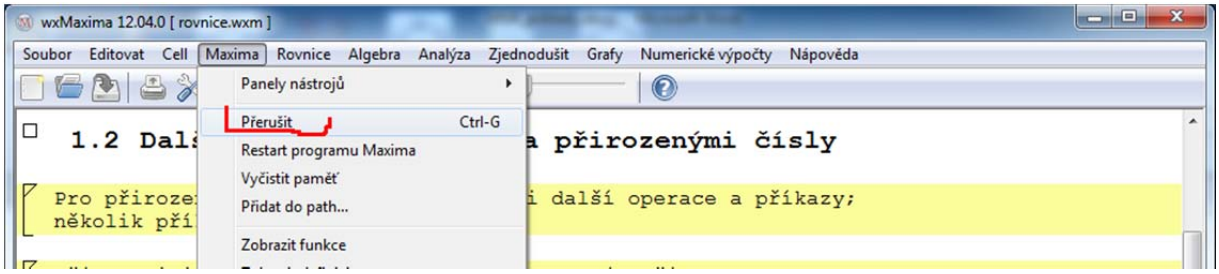


Obr. 2.5

6. S přirozenými čísly můžeme provádět mnoho operací a využívat desítky dalších funkcí, my jsme vybrali jako ukázku jen několik: (obrázek 2.6a)
   divisors(p); příkaz *divisors* vypíše všechny dělitele daného čísla
   integer_partitions(q); příkaz *integer_partitions(n)* či
   integer_partitions(q,2); příkaz *integer_partitions(n,k)* vypíše všechny navzájem různé podmnožiny (případně všechny různé *k*-prvkové podmnožiny) přirozených čísel takové, že součet jejich prvků je roven číslu n. Poznámka: nezkoušejte to pro příliš velká *n*, výpočet bude trvat velmi dlouho a budete muset zachraňovat situaci přerušením výpočtu (obr. 2.6b).
   cardinality(integer_partitions(q)); určení počtu výše zmíněných množin



Obr. 2.6a

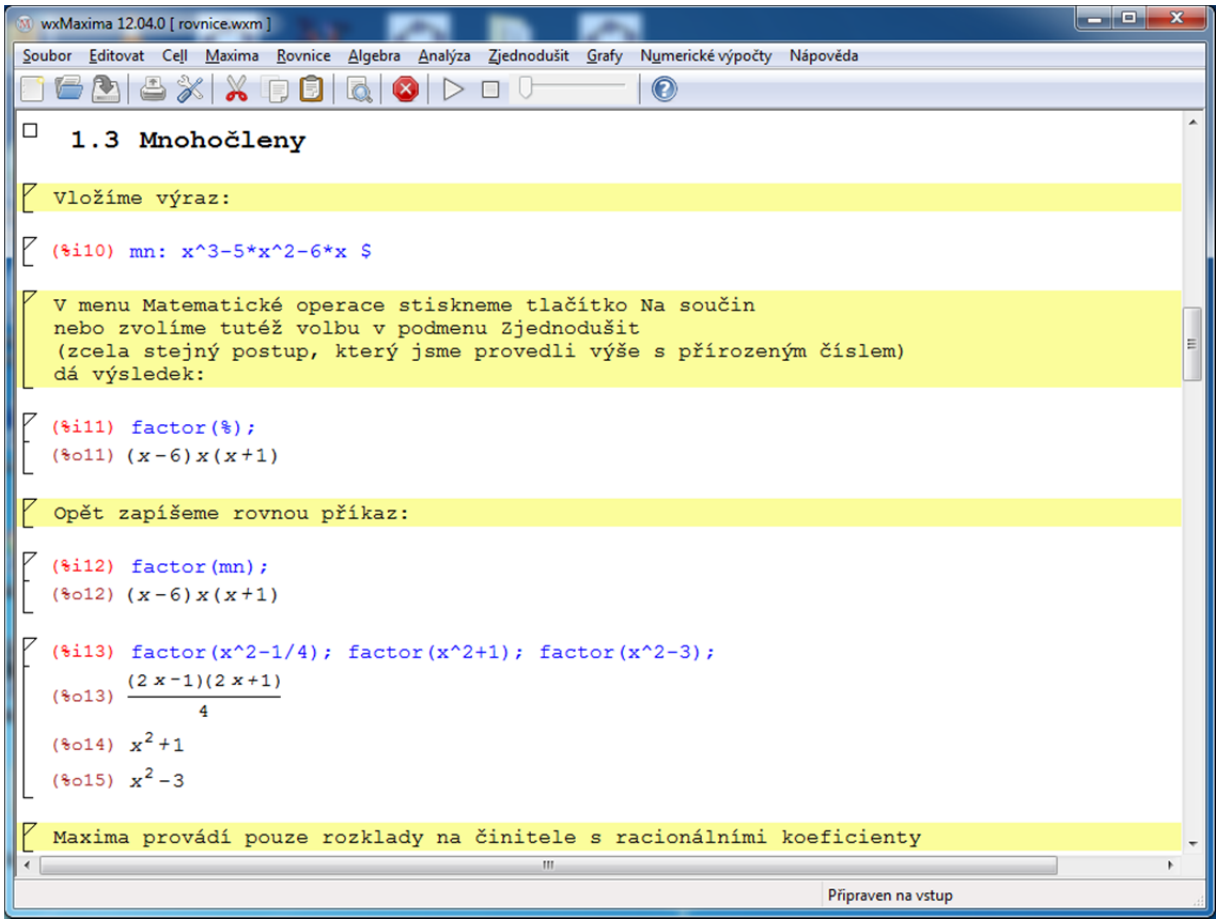


Obr. 2.6b

Rozklad mnohočlenů

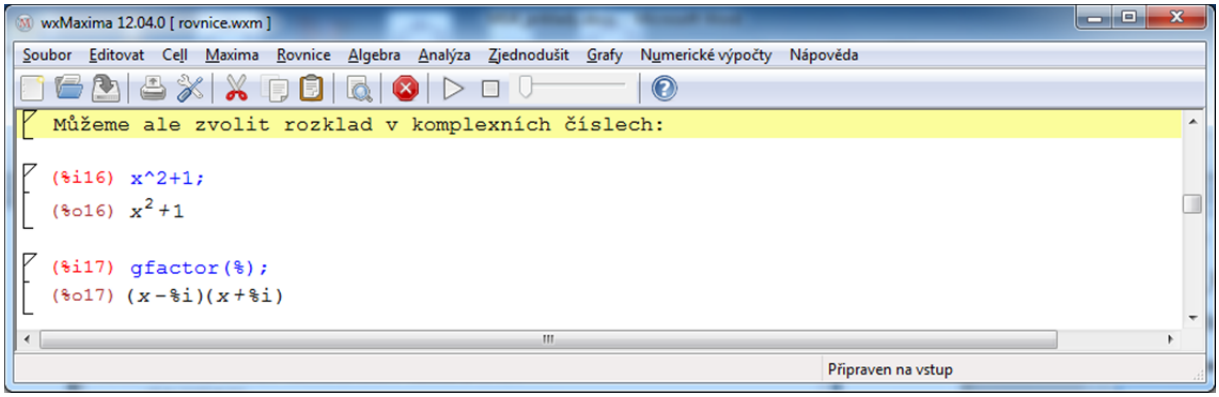
Maxima provádí rozklady na činitele s racionálními koeficienty. Při hledání nulových bodů však můžeme nalézt i hodnoty, které jsou iracionální.

7. Vložíme výraz a opět nejprve rozložíme pomocí příkazu z Panelu nástrojů či z menu. Je to týž příkaz *Na součin*, který jsme použili výše pro přirozené číslo. Víme už, že příkaz má název *factor*.
   Zkusíme postupně přikazy *factor(x^3-5*x^2-6*x); factor(x^2-1/4); factor(x^2+1); factor(x^2-3);* Jejich výstupy ukazují, jaké výrazy (a jak) Maxima rozkládá v součin. (obrázek 2.7)



Obr. 2.7

8. Maxima je ale mnohem silnější nástroj než potřebujeme pro (středo)školskou matematiku, a tak má samozřejmě i nástroje pro rozklad komplexního výrazu (viz obr. 2.8)

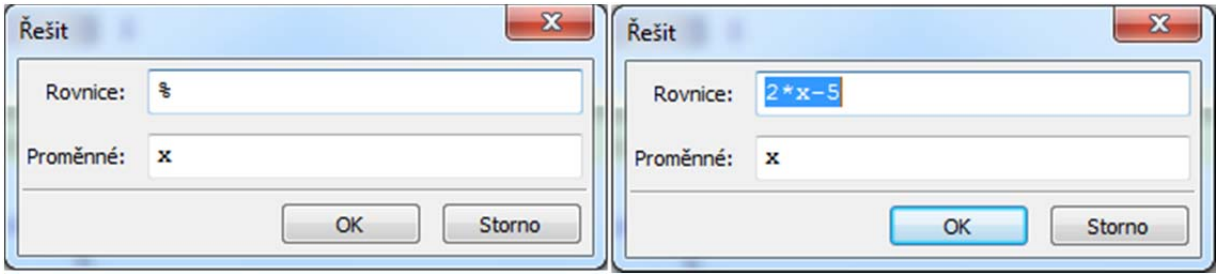


Obr. 2.8

Rovnice

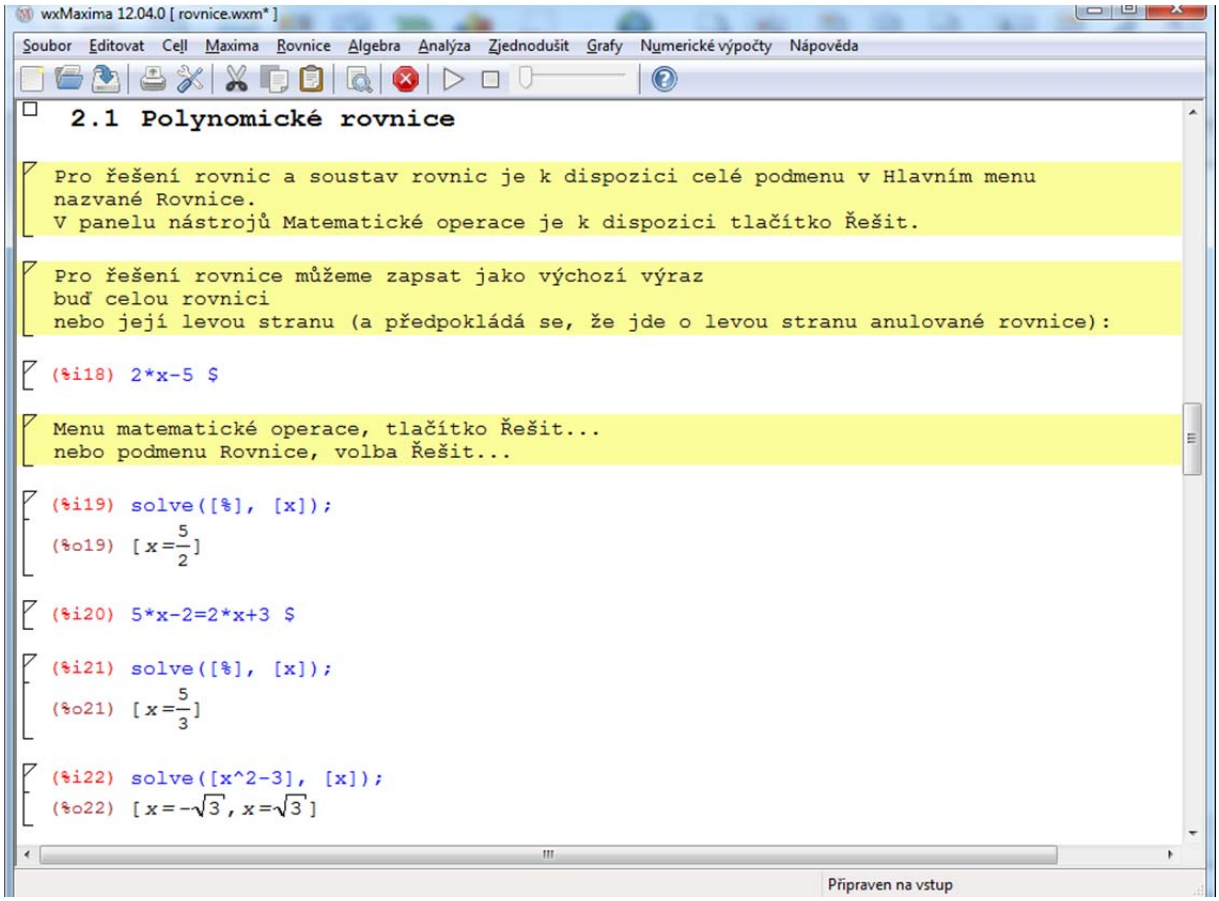
Pro řešení rovnic a soustav rovnic je k dispozici celé podmenu v *Hlavním menu* nazvané *Rovnice*.
V panelu nástrojů *Matematické operace* je k dispozici tlačítko *Řešit…*.
Pro řešení rovnice můžeme zapsat jako výchozí výraz buď celou rovnici, nebo její levou stranu
(a předpokládá se, že jde o levou stranu anulované rovnice):

9. Rovnici zapíšeme buď do vstupního pole a následně vyvoláme dialog (obr. 2.9a), nebo rovnici zapíšeme rovnou do dialogu (obr. 2.9b)



Obr. 2.9a, b

10. Vidíme, že potřebný příkaz má tvar *solve([rovnice], [proměnná]);* tedy například *solve([%], [x]);* nebo *solve([x^2-3], [x]);*
Poslední vstup dává výstup [*x=-sqrt(3),x=sqrt(3)*], příkaz tedy počítá i iracionální kořeny a (není-li řečeno jinak) vyjadřuje je algebraicky. (viz obr. 2.10)

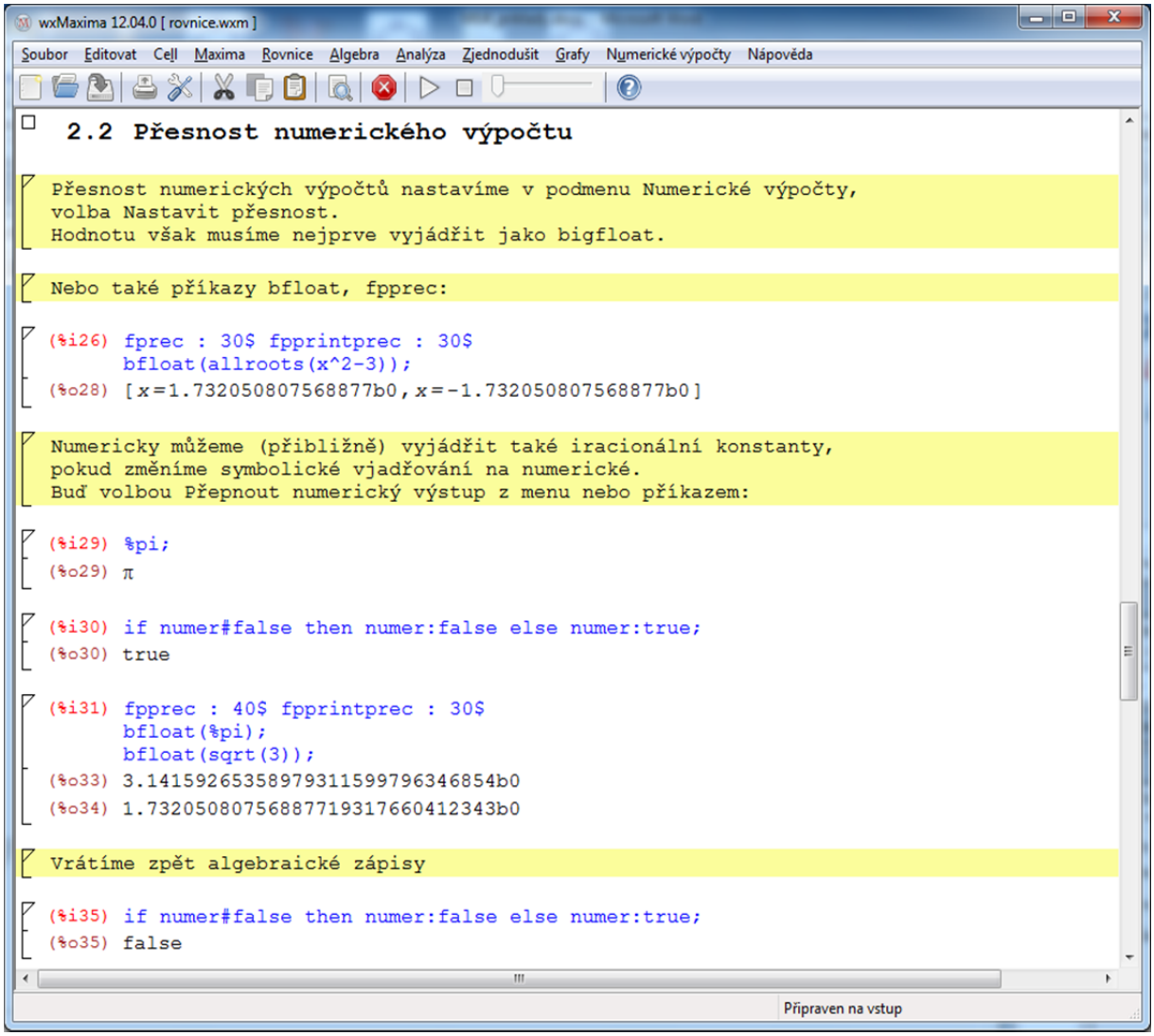


Obr. 2.10

V podmenu *Rovnice* je ale větší nabídka příkazů: příkazy *Kořeny polynomu (reálné), Kořeny polynomu (bfloat) – tedy s nastavitelnou přesností, Kořeny polynomu – s pevnou přesností* a příkaz *Najít kořen*, který dovoluje zadat interval, na němž numericky hledáme kořen.

Přesnost numerického výpočtu

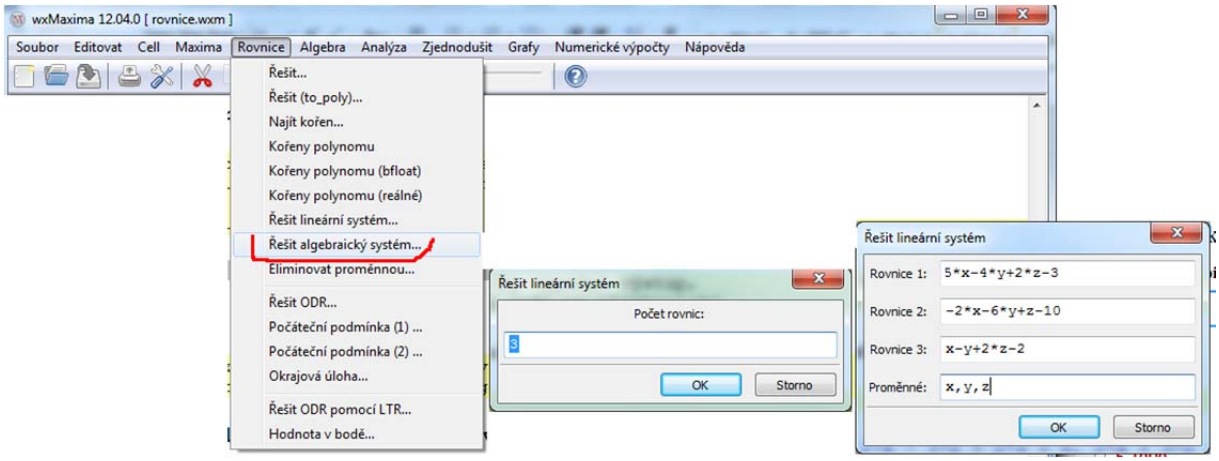
Přesnost numerických výpočtů nastavíme v podmenu *Numerické výpočty*, volba *Nastavit přesnost*. Je však možné ovlivnit přesnost jen takových hodnot, které jsou vyjádřeny jako tzv. bigfloat. Na tento formát hodnotu převedeme volbou *Přesnost bigfloat*.
Další možností, jak docílit téhož, samozřejmě je zapsání příslušných příkazů do vstupního pole: *fprec*, *fpprintprec*, *bffloat*.
Numericky můžeme (přibližně) vyjádřit také iracionální konstanty, pokud změníme symbolické vyjadřování na numerické.
Toho docílíme buď volbou *Přepnout numerický výstup* z menu, nebo příkazem (který se ale po výběru z menu vloží automaticky): *if numer#false then numer:false else numer:true;* (velmi volný „překlad“: není-li vypnuta volba *numer*, tak ji vypni, jinak ji zapni).



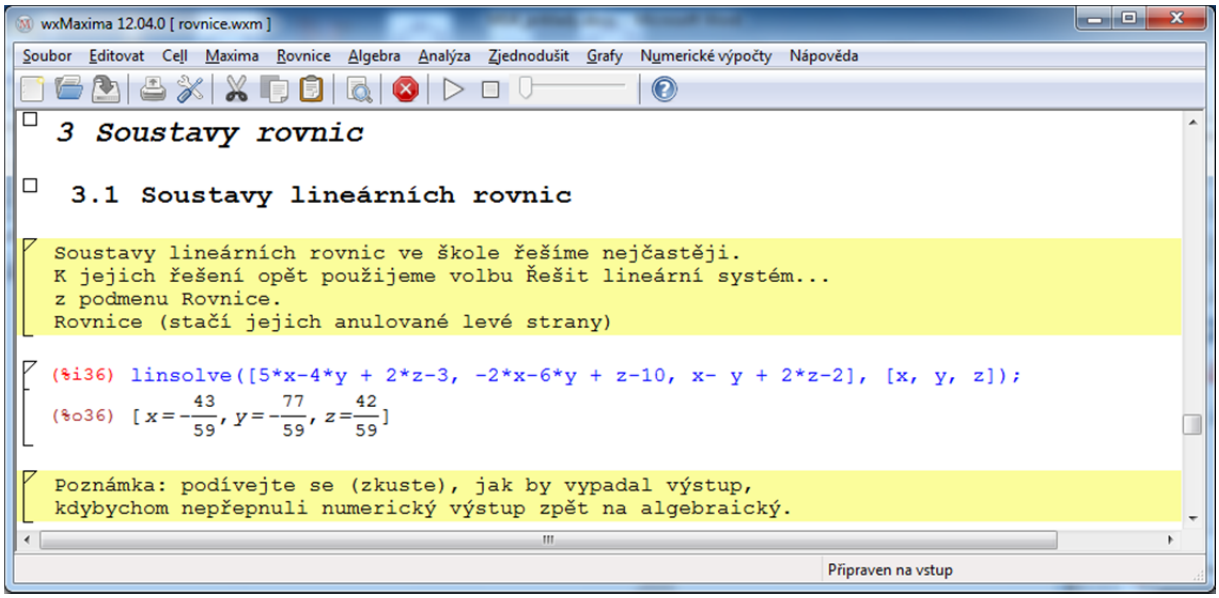
Obr. 2.11

Soustavy rovnic

Soustav lineárních rovnic řešíme pomocí volby *Řešit lineární systém*. Volba vyvolá dialog, kam zapíšeme rovnice (nebo jejich anulované levé strany – obrázek 2.12), nebo odkazy na příslušné vstupní výrazy (proměnné vstupu – např. *%i5*).
Příslušný příkaz má tvar: *linsolve([seznam rovnic], [seznam proměnných]);* tedy např. *linsolve([5*x-4*y + 2*z-3, -2*x-6*y + z-10, x- y + 2*z-2], [x, y, z]);* (viz obr. 2.13)

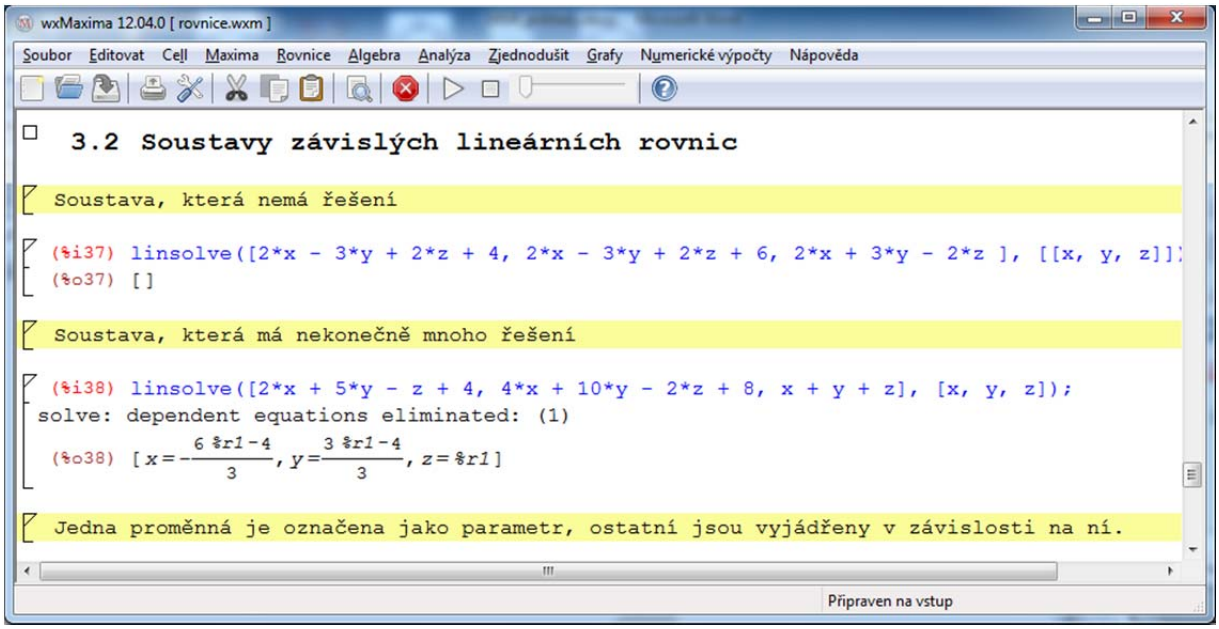


Obr. 2.12



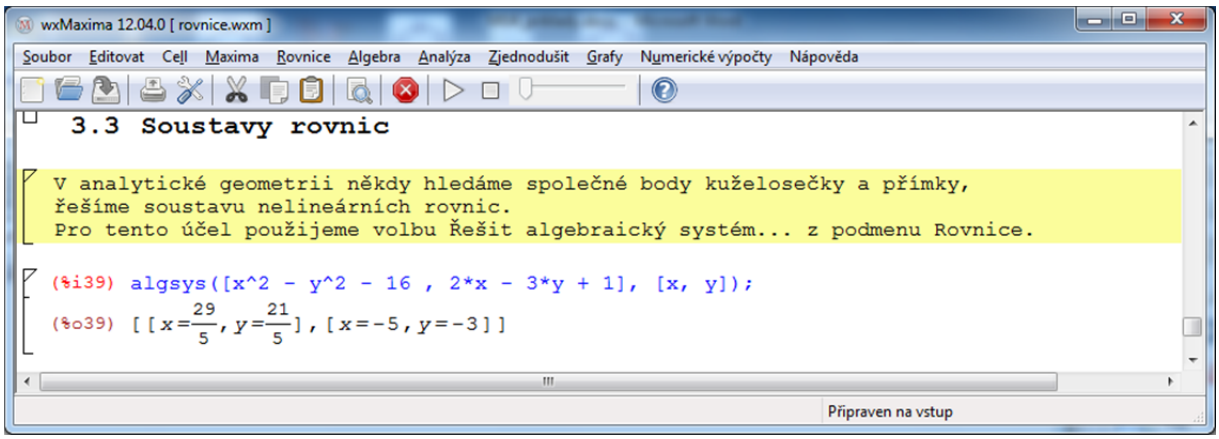
Obr. 2.13

Pokud nemá soustava jediné řešení (jedinou uspořádanou dvojici, trojici, ...), poznáme to ve výstupu: *linsolve([2*x – 3*y + 2*z + 4, 2*x – 3*y + 2*z + 6, 2*x + 3*y – 2*z], [[x, y, z]]);* dává odpověď [ ], soustava nemá řešení,
*linsolve([2*x + 5*y – z + 4, 4*x + 10*y – 2*z + 8, x + y + z], [x, y, z]);* dává odpověď *[x=–(6*%r1–4)/3,y=(3*%r1–4)/3,z=%r1]*, tedy parametrický systém uspořádaných trojic, kdy Jedna proměnná je označena jako parametr a ostatní proměnné jsou vyjádřeny v závislosti na ní (obrázek 2.14).



Obr. 2.14

Pro řešení soustavy algebraických rovnic, které nejsou lineární, je učena volba *Řešit algebraický systém…* ve zmíněném podmenu *Rovnice*. Odpovídá jí příkaz *algsys*, například:
příkaz *algsys([x^2 - y^2 - 16 , 2*x - 3*y + 1], [x, y]);* vrátí jako výstup dvě uspořádané dvojice: *[[x=29/5,y=21/5],[x=-5,y=-3]]* (obrázek 2.15).



Obr. 2.15

Sestrojený list a příslušný html dokument

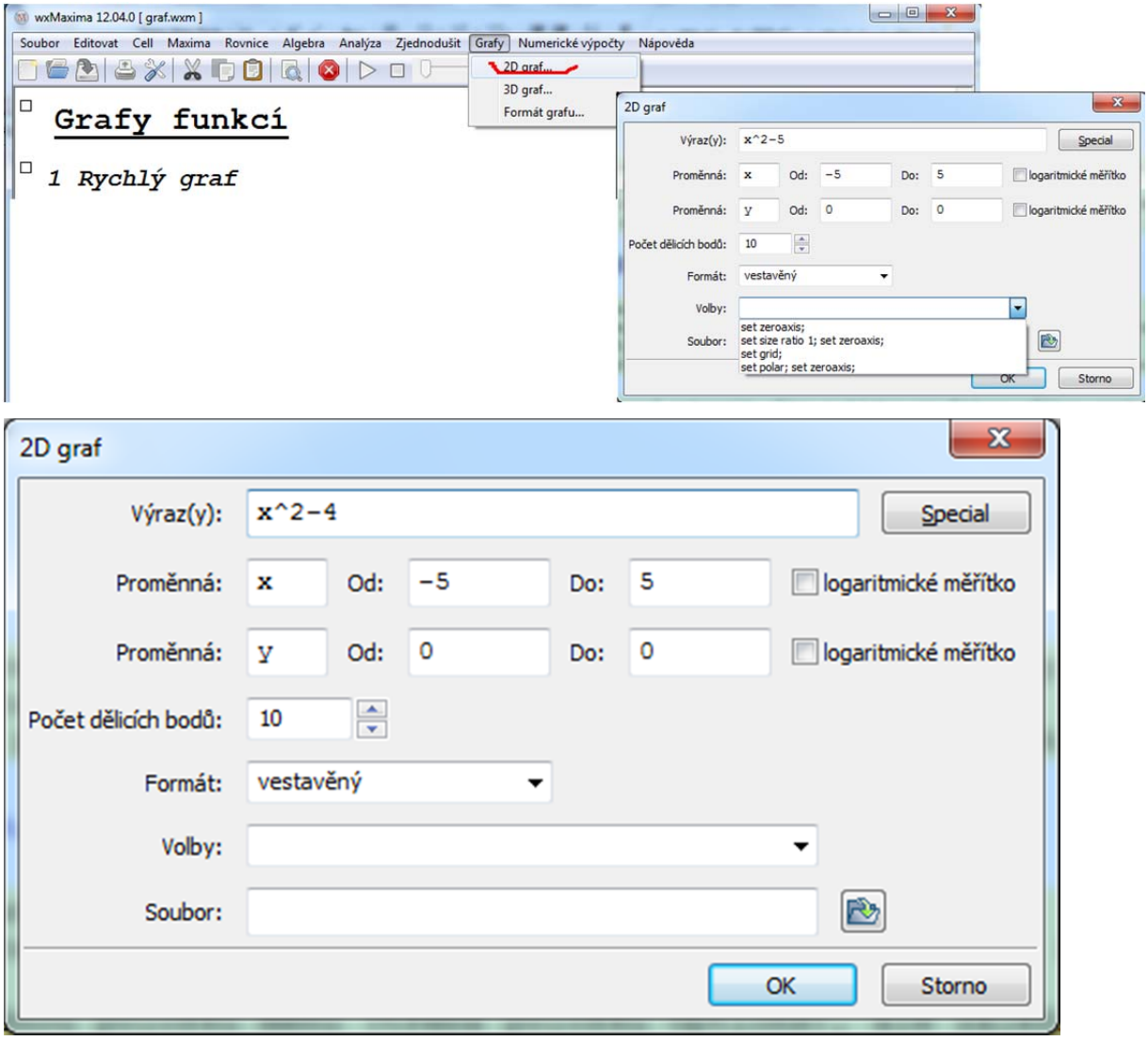
najdete v souborech *rovnice.vwm* a *rovnice.html*. Prvý z nich otevřete – a můžete s ním libovolně pracovat – v programu wxMaxima, druhý jmenovaný soubor otevřete ve webovém prohlížeči.

Příklad třetí – grafy funkcí

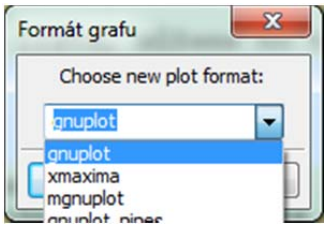
Ve třetím příkladu sestrojíme grafy několika funkcí různými postupy a nakonec si ukážeme použití tzv. posuvníku.
Při konstrukci modelu využijeme postupy uvedené v předcházejících příkladech. Ty už nebudeme tak podrobně popisovat a ilustrovat.
Navíc, celý výklad je – podobně jako v předešlých příkladech – podrobně komentován v přiloženém modelu *graf.wxm*. Pokud ho otevřete a necháte *Vyhodnotit všechna vstupní pole* (volba dostupná z podmenu *Cell*), získáte kompletní výklad. Zde budeme komentovat jen pro daný účel nejpodstatnější příkazy a vlastnosti programu.

Vložení grafu funkce, předpis

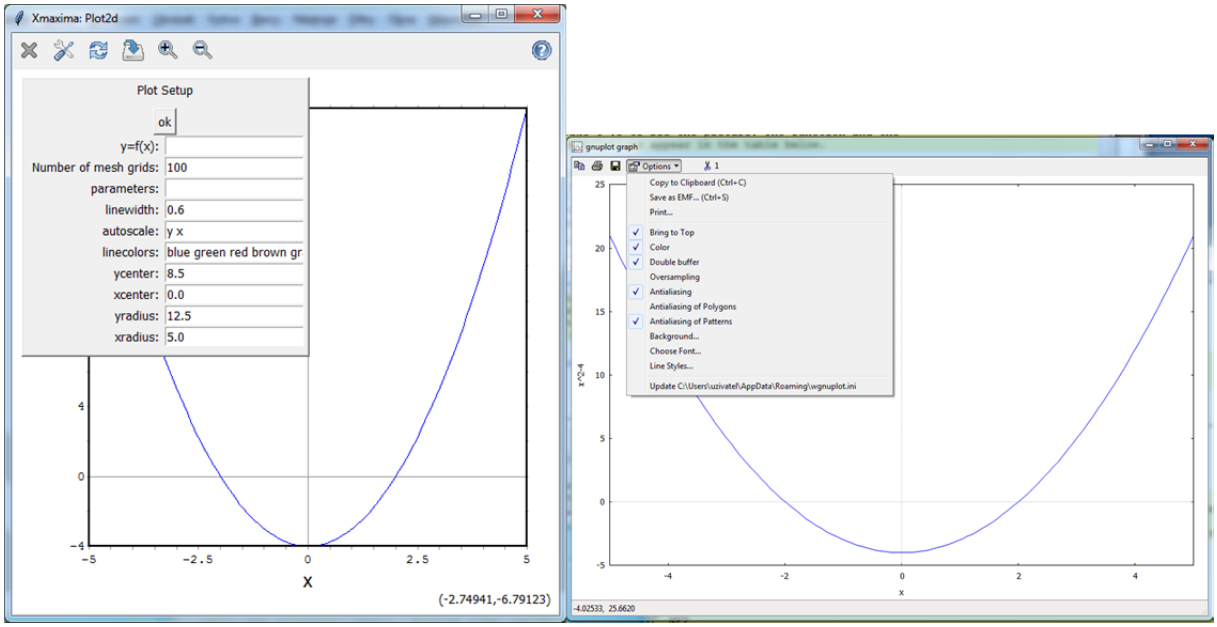
Rychlý graf
Pokud chceme rychle vykreslit graf funkce, stačí využít dialog *2d graf…* z menu *Grafy* a do něj předpis funkce a meze pro nezávisle proměnnou zapsat (obr. 3.1).
Graf se zobrazí v samostatném okně. To má své výhody – okno má vlastní menu a v něm nástroje pro práci s grafem, jeho uložení či kopírování do schránky.
Vzhled okna a možnosti jeho menu však závisí na vybraném formátu grafu. Ten vybíráme v podmenu *Grafy* položkou *Formát grafu* (obr. 3.2, 3.3).
Chceme-li takto sestrojený graf umístit do pracovního listu wxMaxima, můžeme ho tam vložit jako obrázek.



Obr. 3.1



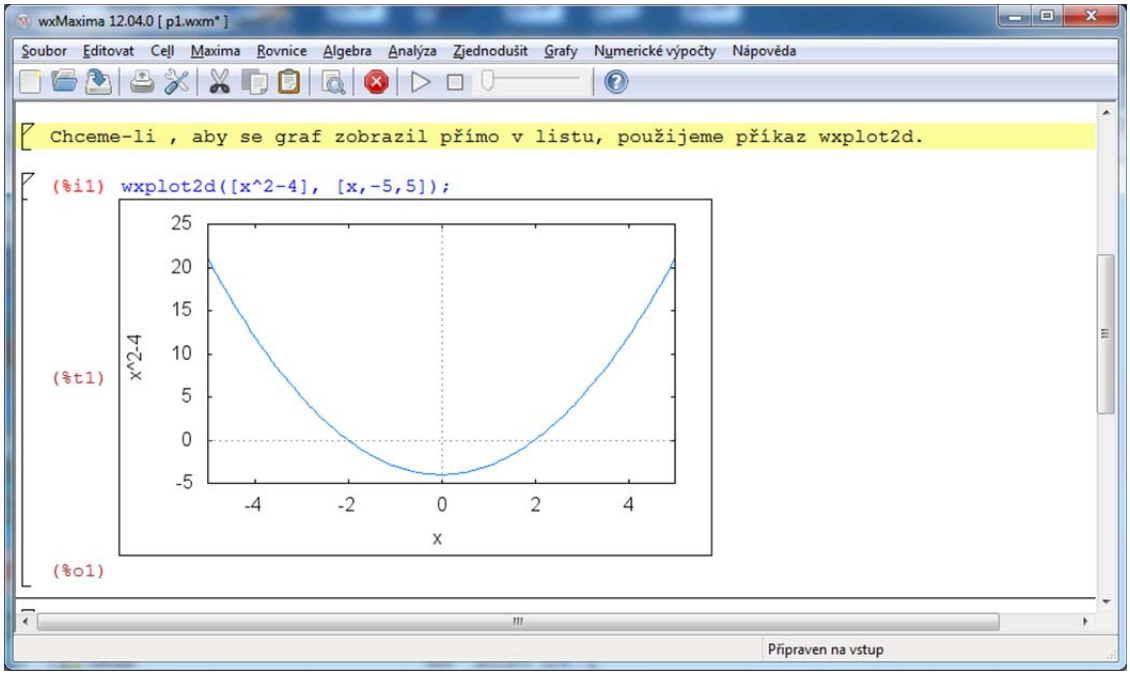
Obr. 3.2



Obr. 3.3

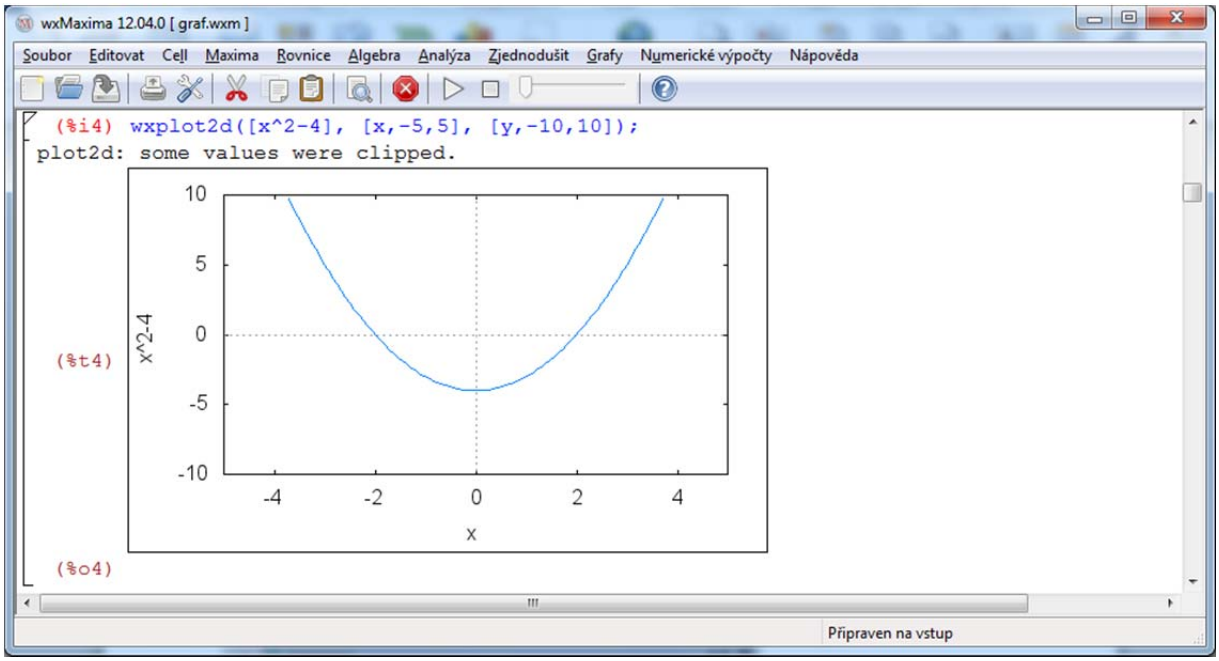
Po provedení zůstane v pracovním listu zapsaný příkaz *plot2d*, který vyvolává výstup grafu do nového okna například : *plot2d([x^2-5], [x,-5,5]);*
Tento příkaz však může mít mnohem více tzv. parametrů, zmíníme se o nich dále.

Chceme-li, aby se graf zobrazil přímo v listu, použijeme příkaz *wxplot2d*. Příklad výstupu vidíme na obrázku 3.4.



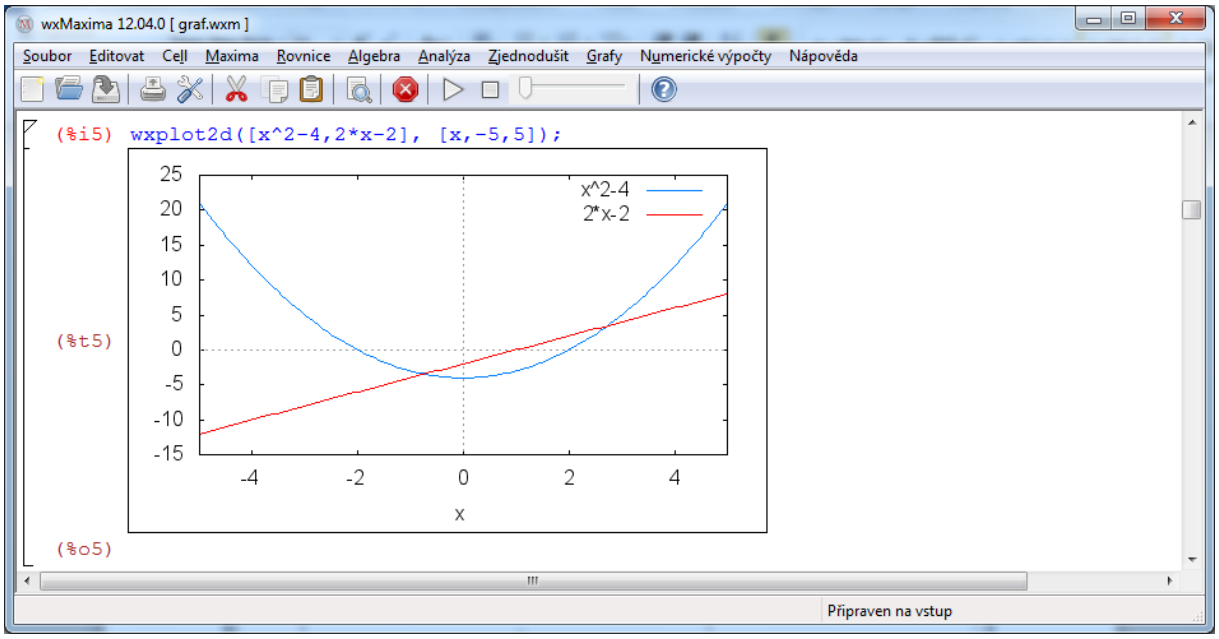
Obr. 3.4

"Základní" kreslicí příkaz Maximy je příkaz *draw* (a *draw2d, draw3d*). Před jeho prvým voláním v daném listu je třeba vyvolat příslušnou knihovnu: *load(draw)$*.
Budeme ale využívat spíše funkcí *plot: plot2d*, případně *plot3d* (či jejich varianty *wxplot…*, které umístí rámeček s grafem přímo do pracovního listu wxMaxima). Pro kreslení grafů ale existuje ještě řada dalších funkcí.
Každý z těchto příkazů má tvar (příklad pro *wxplot2d*): (wx)*plot2d(funkce, rozsah(x), podmínky*). Přitom *podmínek* bývá více a mohou obsahovat i *rozsah(y)*. Určení rozsahu zobrazené části grafu pro osu *y* může vynutit takové zobrazení grafu, kde budou některé hodnoty oříznuté (graf nevyplní celý obdélník) viz obr. 3.5.



Obr. 3.5

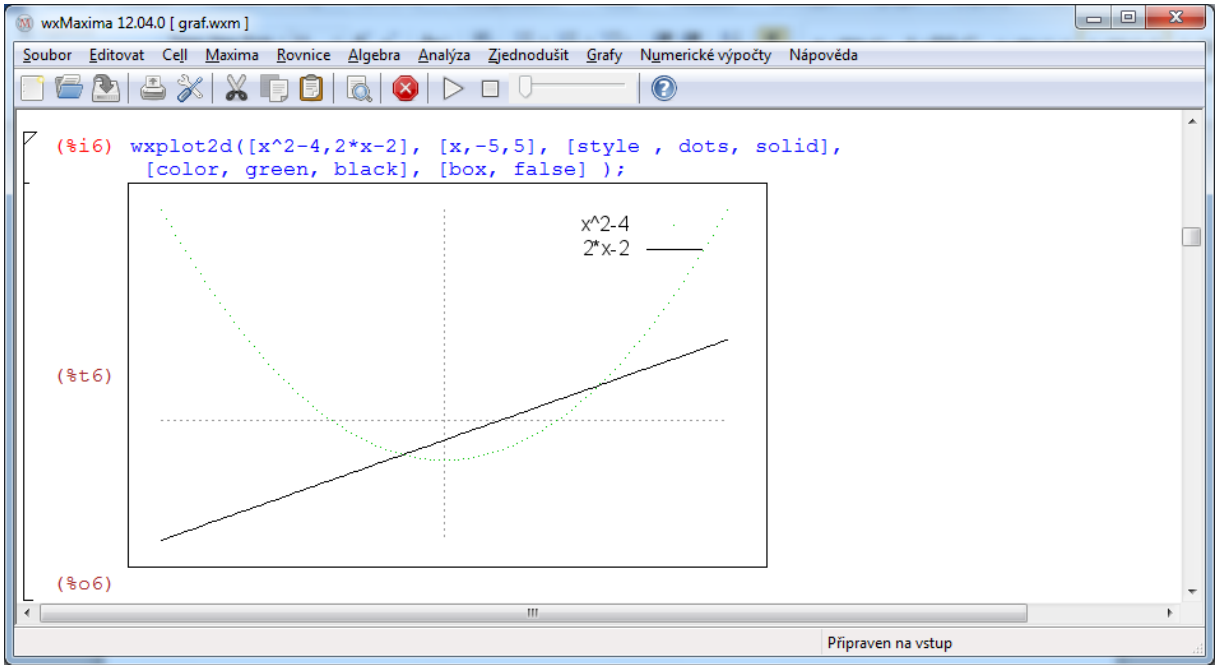
Zobrazení několika grafů
Chceme-li v jednom obrázku zobrazit několik grafů, uvedeme jako prvý parametr kreslicí funkce jejich seznam (v prvé hranaté závorce, proto tam ta závorka je). Zápis a výsledný výstup vidíme na obrázku 3.6.



Obr. 3.6

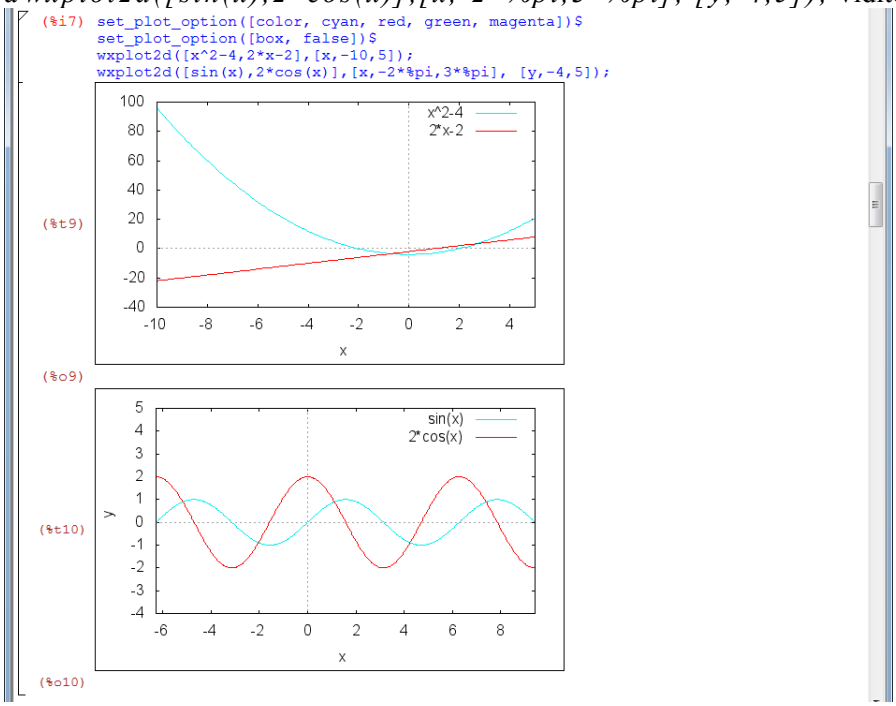
Podmínky vykreslování grafů

V seznamu podmínek nastavujeme další vlastnosti grafu či grafů: barvy, zobrazení os, typy čar, ...
Podívejte se do nápovědy (a do uvedených příkladů).
Každá z podmínek má tvar seznamu (je uzavřena v hranatých závorkách), kde jako prvý stojí název parametru (název podmínky) a za ním jeho hodnota či seznam hodnot, například:
*wxplot2d([x^2-4,2*x-2], [x,-5,5], [style, dots, solid], [color, green, black], [box, false])*; (obr. 3.7)



Obr. 3.7

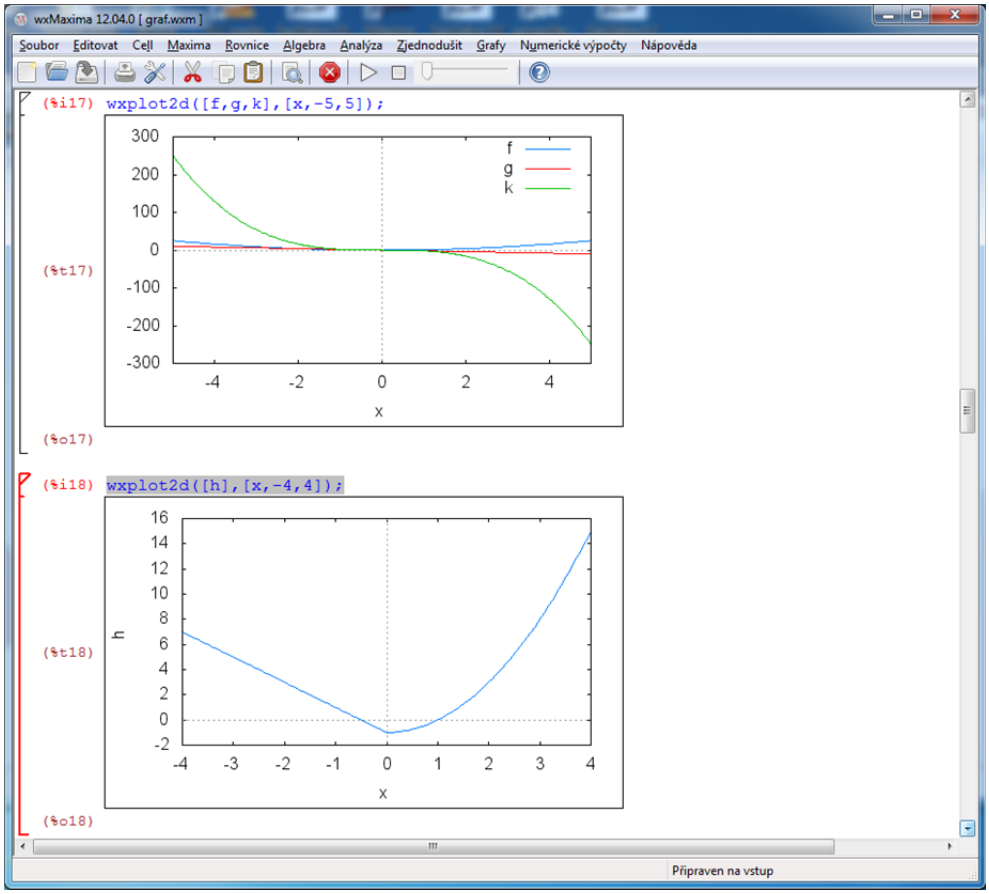
Některé jednotlivé parametry (podmínky) můžeme nastavit globálně pro všechny následující grafy funkcí (příkazem) *set_plot_option* – například:
set_plot_option([color, cyan, red, green, magenta])$
set_plot_option([box, false])$
Následující výstup příkazů *wxplot2d([x^2-4,2*x-2],[x,-10,5]);*
a *wxplot2d([sin(x),2*cos(x)],[x,-2*%pi,3*%pi], [y,-4,5]);* vidíte na obrázku 3.8.



Obr. 3.8

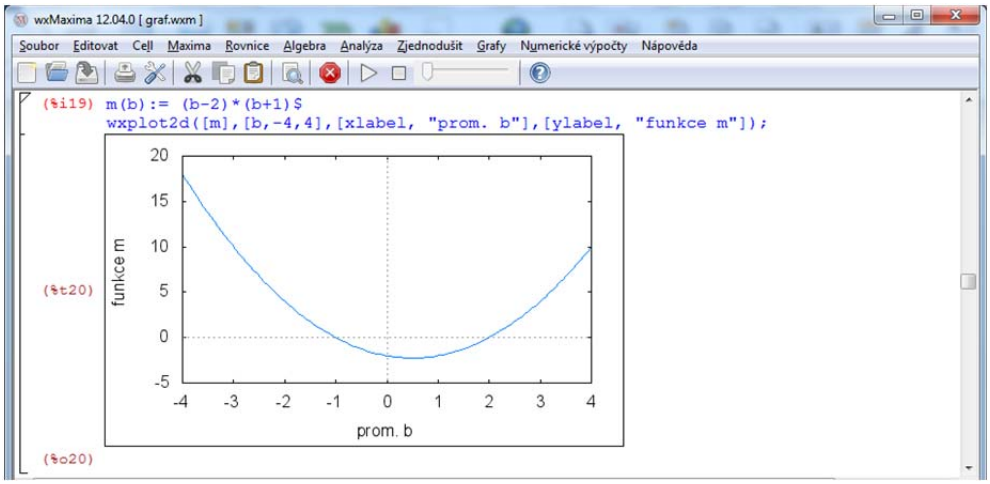
Pojmenování (definice) funkcí

Stejně, jako máme k dispozici pojmenování hodnot proměnnými, můžeme si pojmenovat (tedy definovat) také vlastní funkce. Pro přiřazení však používáme místo dvojtečky symbol (dvojznak) **:=** .
Například: *f(x):= x^2$ g(x):= -2*x$*
Předpis funkce může obsahovat i podmínku: *h(x) := if x < 0 then –2*x – 1 else –1 + x^2 $*
Tedy: pro $x < 0$ je funkce určena předpisem $y = -2*x-1$, pro $x \geq 0$) předpisem $y = -1+x^2$.
S funkcemi můžeme provádět aritmetické operace, například *k(x):= f(x)*g(x);*
A pochopitelně můžeme vykreslovat jejich grafy. Příklad grafů výše definovaných funkcí vykreslených postupně pomocí příkazů *wxplot2d([f,g,k],[x,-5,5]); wxplot2d([h],[x,-4,4]);* vidíme na obrázku 3.9.



Obr. 3.9

Nezávisle proměnná se pochopitelně nemusí jmenovat *x* (obr. 3.10):



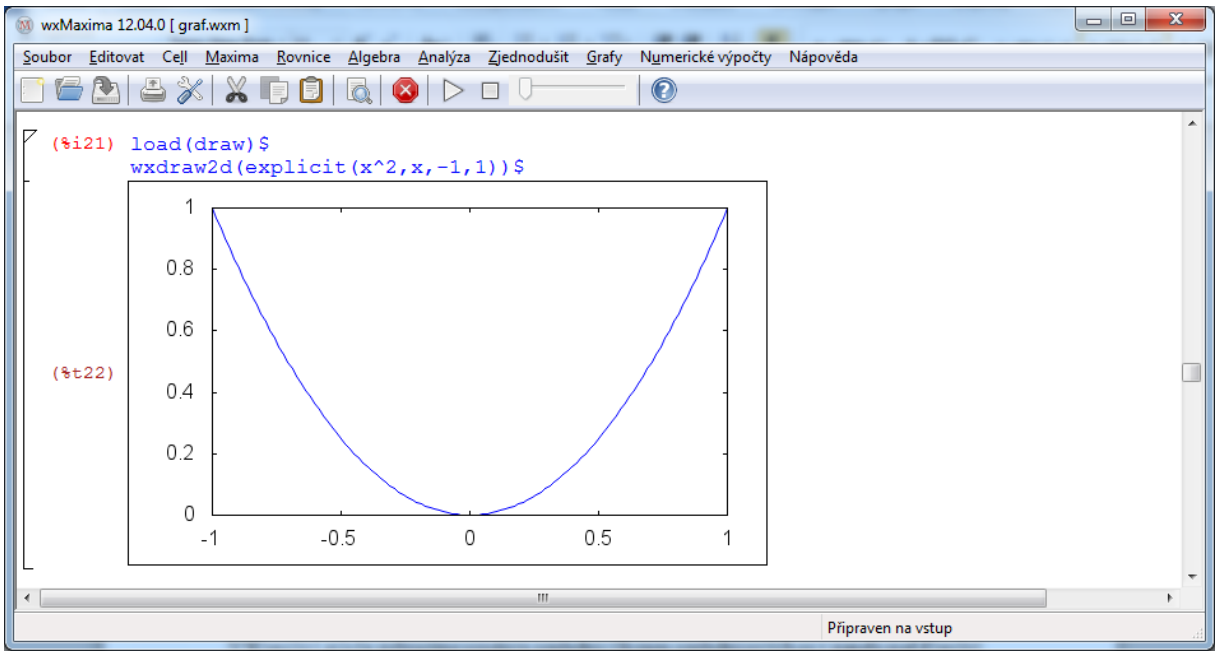
Obr. 3.10

Posuvník

Prostředí wxMaxima (tedy ne program Maxima sám, ale jeho grafická nadstavba) poskytuje užitečný grafický nástroj, který může zlepšit názornost vytvářených materiálů – tzv. *posuvník*, který umožní vytvořit animované grafy.
Pokud ho chceme při vykreslování grafů funkcí (jedné proměnné) využít, musíme místo příkazu *draw2d* volat funkci (příkaz) *with_slider_draw*.
Funkci *draw* jsme dosud nepoužívali, všimněte si jejího zápisu, použití a parametrů: V příkazu *draw2d* (nebo *wxdraw2d*, který vykreslí graf rovnou do okna pracovního listu wxMaxima) se (explicitně zadaná) funkce vykresluje příkazem *explicit*:

```
load(draw)$
wxdraw2d(explicit(x^2,x,-1,1))$
```

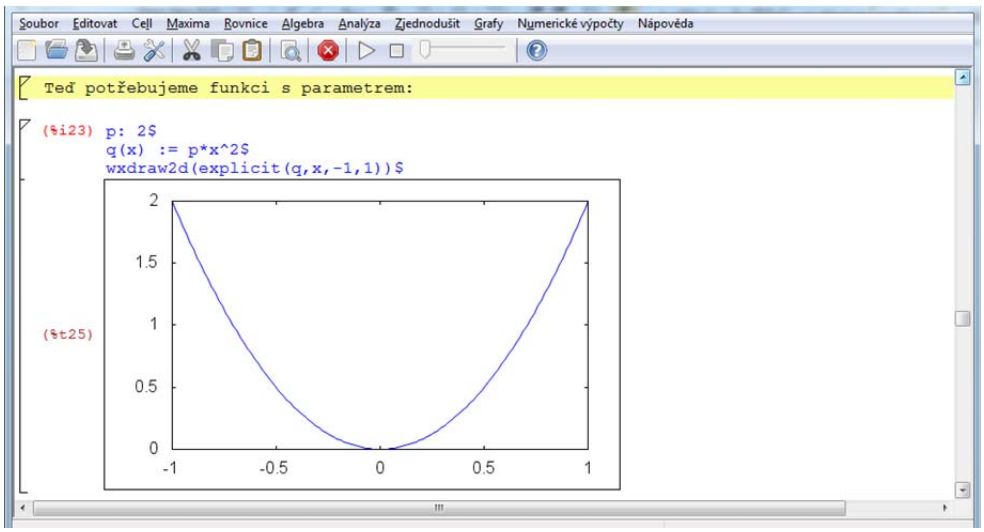
(obr. 3.11)



Obr. 3.11

Pro vykreslení parametrického systému funkcí (ať už s animací, nebo bez ní) potřebujeme definovat funkci s parametrem: (obr. 3.12)

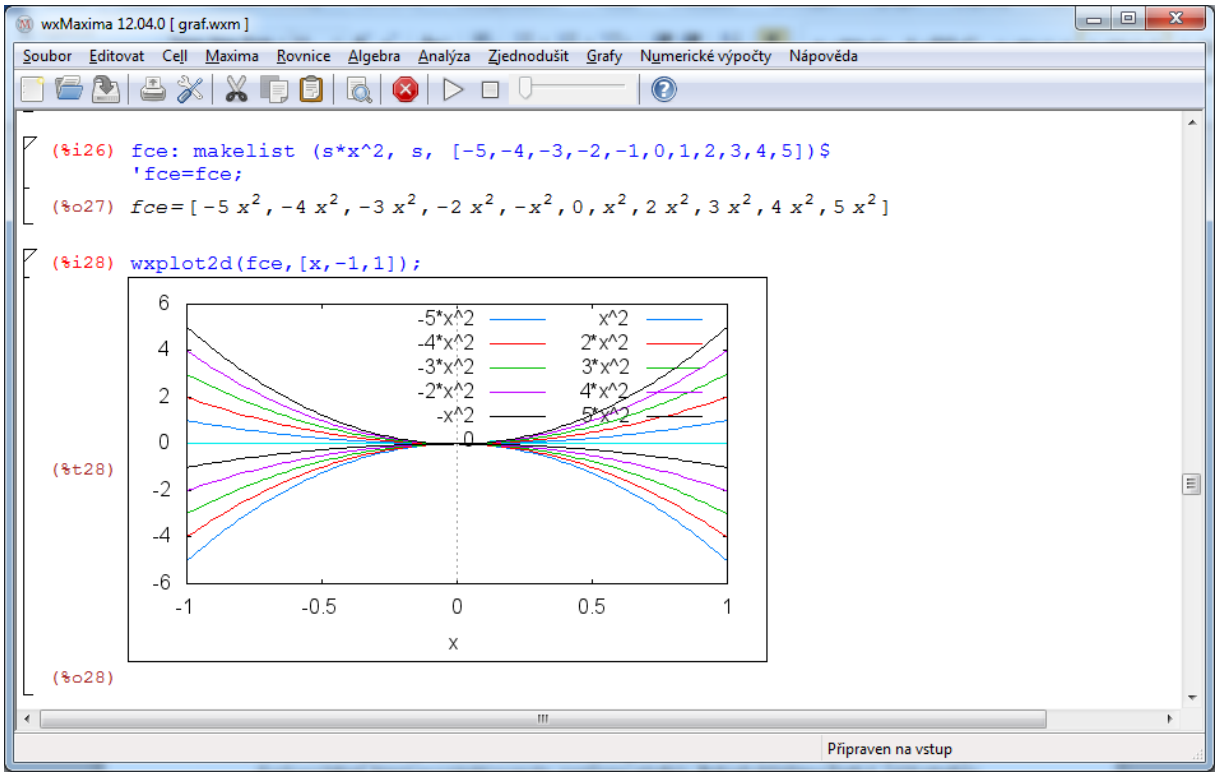
```
p: 2$                          parametr p s definovanou hodnotou
q(x) := p*x^2$
wxdraw2d(explicit(q,x,-1,1))$
```



Obr. 3.12

Graf ale zobrazí pouze funkci pro aktuální hodnotu parametru (obr. 3.12). Parametr musíme vybírat z připraveného seznamu několika hodnot.
Takhle připravíme všechny funkce s parametrem ze seznamu:
*fce: makelist (s*x^2, s, [-5,-4,-3,-2,-1,0,1,2,3,4,5])$* příprava seznamu
'fce=fce; ... a výstup:
*fce=[-5*x^2,-4*x^2,-3*x^2,-2*x^2,-x^2,0,x^2,2*x^2,3*x^2,4*x^2,5*x^2]*

Výstup příkazu *wxplot2d(fce,[x,-1,1]);* vidíme na obrázku 3.13:



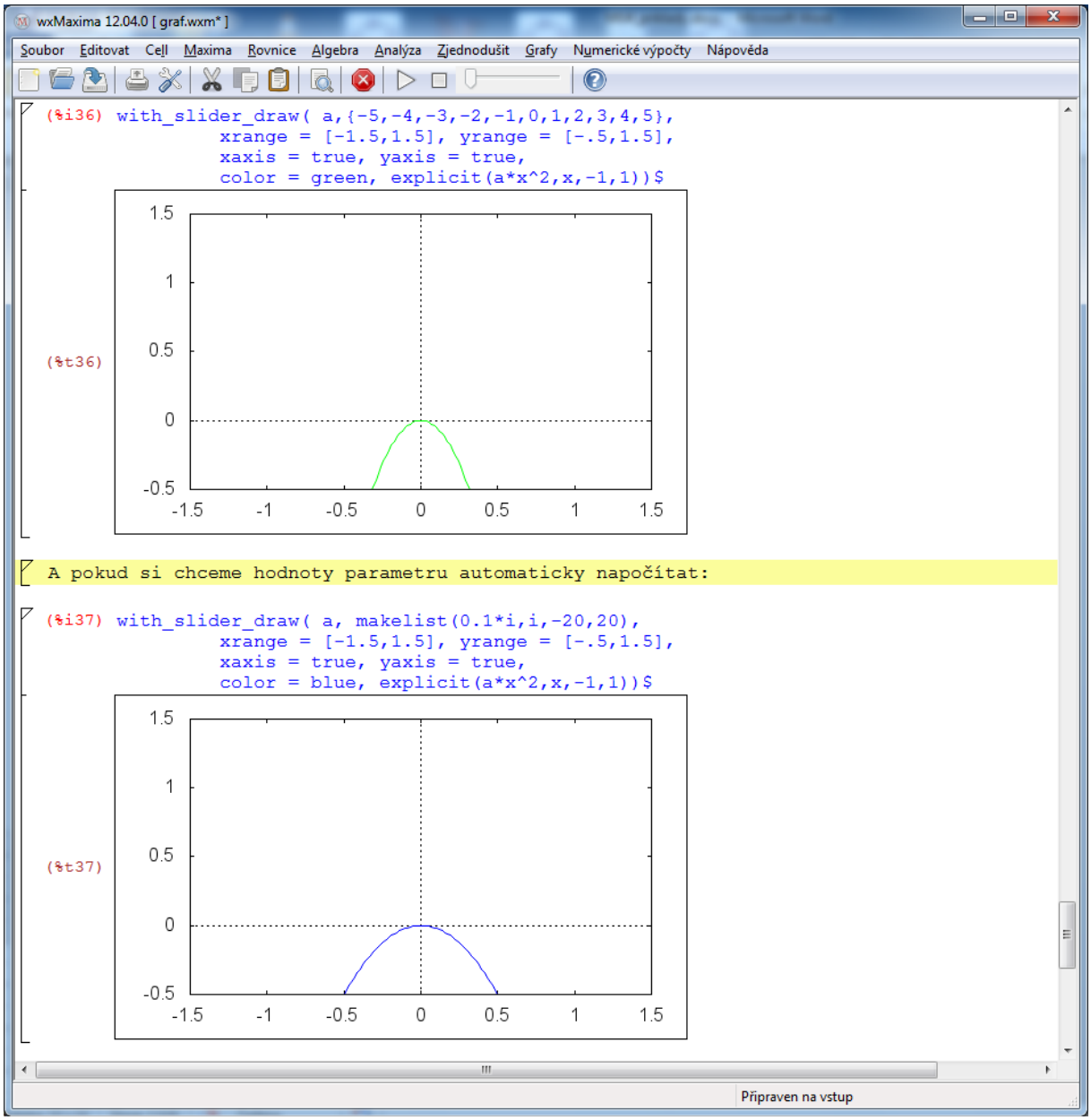
Obr. 3.13

Teď jsou všechny funkce vykresleny v jediném obrázku. Pokud bychom chtěli, aby se vykreslily postupně, do samostatných grafů, můžeme je nechat vykreslit pomocí příkazu cyklu:
for d:-1 thru 1 step 0.5 do
*wxdraw2d(xrange = [-0.5,1.5], yrange = [-1,2], explicit(d*x^2,x,0,1));*
Výstupem bude 5 obrázků, podívejte se na ně do přiloženého souboru *graf.wxm*.

My bychom však chtěli, aby se grafy objevovaly jako animace, postupně v jednom obrázku. Toho lze docílit právě pomocí posuvníku a s ním svázaného příkazu *with_slider_draw*, který jako své parametry obsahuje – mimo jiné – název parametru, množinu jeho hodnot a v příkazu *explicit* předpis funkce a její definiční obor:
Příkaz *with_slider_draw(a,{-5,-4,-3,-2,-1,0,1,2,3,4,5}, explicit(a*x^2,x,-1,1))$* vykreslí graf funkce $y = a*x^2$ na intervalu $<-1, 1>$ postupně pro hodnoty parametru a z uvedené množiny.
Po zadání příkazu nechte Maximu počítat (obrázek bude možná trochu problikávat). Poté klikněte do obrázku a otáčejte kolečkem myši, nebo klikejte na ikonu posuvníku – na trojúhelník na *Panelu nástrojů,* nebo za čtvereček na tomto panelu táhněte do strany. Graf se bude animovat.

Efekt však není takový, jaký bychom očekávali. Každý graf se totiž vykresluje tak, aby optimálně vyplnil vymezený prostor, tedy mění se měřítko na ose y, a změny grafu vzhledem k hodnotě parametru nejsou tudíž zřetelné. Automatické změně měřítka zabráníme nastavením pevného rozsahu osy y v parametrech příkazu:
with_slider_draw(a,{-5,-4,-3,-2,-1,0,1,2,3,4,5},
*xrange = [-1.5,1.5], yrange = [-.5,1.5],explicit(a*x^2,x,-1,1))$.*

Na další možnosti se podívejte do souboru nebo na obrázek 3.14.



Obr.3.14

Sestrojený list

najdete v souboru *graf.wxm*. K tomuto příkladu jsme nevytvářeli html stránku.