Základní list

Úplný základ - načítání knihoven, informace, restart

To, že čtete tento list znamená, že jste úspěšně zvládli spuštění Maplu a načtení listu do něj ;-). Pokud jste úplní začátečníci, chcete se něco dozvědět o Maplu a máte dost času můžete si pustit Starter tour (nalézá se jen v novějších verzích). Najdete jej v nabídce Help - Contents (pokud je samozřejmě ve Vaší verzi obsažena). Jinak doporučuji kurz RNDr Pavla Pyriha - odkaz na něj najdete na stránce, kde jste stáhli tento list.

Základní informace

Standartní funkce se zapisují v Maplu přirozeným způsobem - notací, používanou ve většině programech, kde se vyskytuje matematika. Stačí je psát tak, jak se například používají v programovacím jazyce Pascal.

Proto uvedu pouze způsob zadávání exponenciely a logaritmu :

```
> exp(x);
> log(x)=ln(x);
```

$$e^{x}$$

$$\ln(x) = \ln(x)$$

A takhle vypadá absolutní hodnota a druhá odmocnina:

```
> abs(-1);
> sqrt(4);
```

$$1$$

$$2$$

Jak už by mělo být z předešlého zřejmé, každý příkaz je v Maplu nutné ukončit středníkem. Další možnost je ukončit ho dvojtečkou, což určuje, že Maple neprovede žádný výstup na obrazovku.

```
> exp(x):
```

Jak dostat z Maplu další informace

Kromě systému nápovědy, který je mimochodem moc pěkný, existují v Maplu i další příkazy, které vám dají další informace o výpočtech a jejich průběhu.

Nejzajímavějším takovým příkazem je infolevel. Nastavuje se pomocí přirozených čísel 1-5. 1 znamená nejméně informací, 5 nejvíce. Základní nastavení je 1. Infolevel se používá takto :

```
> infolevel[int]:=5;
```

$$infolevel_{int} := 5$$

Takto jsme právě zvýšili informace o průběhu výpočtů integrálu na 5.

```
> Int(sin(x)/x, x=0..infinity)=int(sin(x)/x, x=0..infinity);
int:   Beginning integration with _EnvContinuous=_EnvContinuous, _EnvAll
Solutions=_EnvAllSolutions, and _EnvCauchyPrincipalValue=_EnvCauchyPrinc
ipalValue.
Definite Integration:   Integrating expression on x=0..infinity
Definite Integration:   Using the integrators [Distribution, Piecewise,
Series, O, Polynomial, Ln, LookUp, Cook, Ratpoly, Elliptic, EllipticTrig
, MeijerGSpecial, Improper, Asymptotic, FTOC, MeijerG, Contour]
Definite Integration:   Trying method Distribution.
Definite Integration:   Finished method Distribution. Time elapsed: 0.01
6000s.
Definite Integration:   Trying method Piecewise.
Definite Integration:   Finished method Piecewise. Time elapsed: 0.00000
0s.
Definite Integration:   Trying method Series.
Definite Integration:   Finished method Series. Time elapsed: 0.000000s.
Definite Integration:   Trying method O.
Definite Integration:   Finished method O. Time elapsed: 0.000000s.
Definite Integration:   Trying method Polynomial.
Definite Integration:   Finished method Polynomial. Time elapsed: 0.0000
00s.
Definite Integration:   Trying method Ln.
Definite Integration:   Finished method Ln. Time elapsed: 0.000000s.
Definite Integration:   Trying method LookUp.
IntegralTransform LookUp Integrator:   Integral might be a fouriersin tr
ansform with expr=1/x and s=1.
LookUp Integrator:   the specified integral found with the IntegralTrans
form look up method.
Definite Integration:   Finished method LookUp. Time elapsed: 0.922000s.
Definite Integration:   Method LookUp succeeded.
Definite Integration:   Finished sucessfully.
```

$$\int_0^{\infty} \frac{\sin(x)}{x}\,dx = \frac{\pi}{2}$$

Takhle tedy Maple počítá integrál $\int_0^{\infty} \frac{\sin(x)}{x}\,dx$;-).

Načítání knihoven

Většina potřebných věcí se v Maplu ukrývá v různých knihovnách. Tyto knihovny se zpřístupní teprve po načtení. Toto řešení je dobré zejména vzhledem k obsáhlosti některých knihoven a způsobu jakým je to do nich uloženo. Pokud tedy počítáte s grupami nepotřebujete bředit paměť počítače knihovnou s numerickými funkcemi.

Jednou z nejdůležitějších knihoven je knihovna Student obsahující základní funkce

potřebné k počítání příkladů z okruhu analýzy a kalkulu obecně. Tato knihovan (nebo alespoň většina, jejích funkcí - záleží na instalaci Maplu) se načítá automaticky po spuštění. Pro zábavu a obecné pobavení si ji načteme znova.

```
> with(student);
```

[D, *Diff*, *Doubleint*, *Int*, *Limit*, *Lineint*, *Product*, *Sum*, *Tripleint*, *changevar*, *completesquare*, *distance*, *equate*, *integrand*, *intercept*, *intparts*, *leftbox*, *leftsum*, *makeproc*, *middlebox*, *middlesum*, *midpoint*, *powsubs*, *rightbox*, *rightsum*, *showtangent*, *simpson*, *slope*, *summand*, *trapezoid*]

Podobnou práci na jiné brdo učiní i příkaz readlib.

```
> readlib(student);
```

table([*intercept* = *student/intercept*, *completesquare* = *student/cmpltsq*, *summand* = *student/summand*, *rightsum* = *student/rightsum*, *Int* = *student/Int*, *leftbox* = *student/leftbox*, *rightbox* = *student/rightbox*, *powsubs* = *student/powsubs*, *Lineint* = *student/lineint*, *simpson* = *student/simpson*, *Sum* = *student/Sum*, *equate* = *student/equate*, *slope* = *student/slope*, *middlebox* = *student/middlebox*, *makeproc* = *student/makeproc*, *changevar* = *student/changevar*, *showtangent* = *student/showtngt*, *trapezoid* = *student/trapezoid*, *Limit* = *student/Limit*, *Product* = *student/Product*, *Doubleint* = *student/doubleint*, *distance* = *student/distance*, *integrand* = *student/integrand*, *midpoint* = *student/midpoint*, *Tripleint* = *student/tripleint*, D = D, *leftsum* = *student/leftsum*, *middlesum* = *student/middlesum*, *Diff* = *student/Diff*, *intparts* = *student/intparts*])

Můžeme také načítat jen jednotlivé příkazy, bez toho abychom načetli celou knihovnu.

```
> with(student, rightsum);
```

[*rightsum*]

Restart

Občas je výhodné vynulovat celou paměť, zbavit se zadefinovaných proměnných, funkcí i načtených knihoven. Pokud to chceme udělat stačí napsat:

```
> restart;
```

Další zajímavé operátory a funkce

Upravujeme výrazy

Funkce simplify zjednodušuje výrazy. Provede krácení a vytýkání, umí i základní vzorce pro fce sin a cos.

```
> (x^3-1)/(x-1)*(sin(x)^2-1)/cos(2*x)=simplify((x^3-1)/(x-1)
  *(sin(x)^2-1)/cos(2*x));
```

$$\frac{(x^3-1)\,(\sin(x)^2-1)}{(x-1)\cos(2\,x)} = -\frac{\cos(x)^2\,(x^2+x+1)}{\cos(2\,x)}$$

Můžeme i určovat jaké úpravy se budou provádět viz. help na simplify.

Podobné věci na trochu jiné brdo dělá funkce combine.

```
> Limit(f(x),x=a)*Limit(g(x),x=a)=combine(Limit(f(x),x=a)*Li
  mit(g(x),x=a));
```

$$\left(\lim_{x\to a} f(x)\right)\left(\lim_{x\to a} g(x)\right) = \lim_{x\to a} f(x)\,g(x)$$

Opakem combine je expand:

```
> (x+1)^4=expand((x+1)^4);
```

$$(x+1)^4 = x^4+4\,x^3+6\,x^2+4\,x+1$$

Vyčíslujeme

Pro přibližné vyčíslení výrazu používáme funkci evalf.

```
> Pi = evalf(Pi);
> Pi = evalf(Pi, 20);
```

$$\pi = 3.141592654$$

$$\pi = 3.1415926535897932385$$

Opět evalf je poměrně variabilní příkaz, více o jeho vatriantách viz. help na evalf .

Na určení přesnosti můžeme použít, buď parametr v evalf, nebo globální parametr digits, který je standartně nastaven na 10.

```
> Digits:=20;
> e = evalf(exp(1));
```

$$Digits := 20$$

$$e = 2.7182818284590452354$$

Rozkládáme

Zajímavou funkcí je factor. Pomůže nám faktorizovat polynomy.

```
> x^4+7*x^3+17*x^2+17*x+6 = factor(x^4+7*x^3+17*x^2+17*x+6);
```

$$x^4 + 7\,x^3 + 17\,x^2 + 17\,x + 6 = (x+3)\,(x+2)\,(x+1)^2$$

Rozklad se provádí pouze, lze-li rozložit celá čísla. Pokud to nejde, můžeme k tomu Maple donutit.

```
> factor(x^3+3);
> factor(x^3+3,3^(1/3));
```

$$x^3 + 3$$

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

Můžeme provádět i prvočíselný rozklad přírozených čísel.

```
> ifactor(110160);
```

$$(2)^4\ (3)^4\ (5)\ (17)$$

Faktorizace na prvočísla není samozřejmě vždy ideální, ale lze ji provádět podle více metod viz help na ifactor .

Více o práci s polynomy a faktorizaci viz list Algebra .

Skládáme na více způsobů

Nejdříve si ukážeme skládací operátor. Uplatníme ho později, např. při derivacích.

```
> (sin@cos)(x);
```

$$\sin(\cos(x))$$

Můžeme použít, i dvojitý skládací operátor. Určuje kolikrát složíme funkci samu se sebou.

```
> (sin@@(-1))(x);
> (sin@@(3))(x);
```

$$\arcsin(x)$$

$$(\sin^{(3)})(x)$$

Teď přijde na řadu jedna moc užitečná věc, která ušetří spoustu práce. Operátor $:

```
> x := [i $ i=1..10];
```

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

Pomocí tohoto operátoru můžeme vytvářet skupiny čísel a z těch pak třeba počítat funkční hodnoty a pod. S tímto operátorem lze dělat i další fígle. Čísla například mohou být i necelá, skoky jsou opět po jedné:

```
> z := {k $ k= 1/9..5};
```

$$z := \{\frac{1}{9}, \frac{10}{9}, \frac{19}{9}, \frac{28}{9}, \frac{37}{9}\}$$

Operátor $ je nadán i jistou mírou vlastní "chytrosti".

```
> x := 5$4;
```

$$x := 5, 5, 5, 5$$

Při vytváření jmen si vystačíme i jen s pouhou . (tečkou ;-)).

```
> x := p.(5..10);
```

$$x := p.(5 .. 10)$$

Tak to je operátor, ale na podobné věci máme navíc i funkci, jmenuje se seq a je moc dobrá. Pomocí ní třeba konečně zjistíme jak, že je to s těmi siny ;-).

```
> seq(sin(Pi*i/4), i=0..8);
```

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

Pěkné. Ale pomocí jednoho příkazu navíc - nops, který vrací počet prvků v seznamu to ještě vylepšíme.

```
> M := [seq(i/4, i=0..8)];
> [t,sin(t)]=[seq( [M[i],sin(M[i]*Pi)], i=1..nops(M))];
```

$$M := \left[0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, \frac{5}{4}, \frac{3}{2}, \frac{7}{4}, 2\right]$$

$$[t, \sin(t)] = \left[[0, 0], \left[\frac{1}{4}, \frac{\sqrt{2}}{2}\right], \left[\frac{1}{2}, 1\right], \left[\frac{3}{4}, \frac{\sqrt{2}}{2}\right], [1, 0], \left[\frac{5}{4}, -\frac{\sqrt{2}}{2}\right], \left[\frac{3}{2}, -1\right], \left[\frac{7}{4}, -\frac{\sqrt{2}}{2}\right], [2, 0]\right]$$

Můžeme dokonce přímo volat prvky v našem seznamu.

```
> M_na_druhou := [seq( i^2,i=M)];
```

$$M\_na\_druhou := \left[0, \frac{1}{16}, \frac{1}{4}, \frac{9}{16}, 1, \frac{25}{16}, \frac{9}{4}, \frac{49}{16}, 4\right]$$

Dostali jsme tedy možnost používat v Maple výhody cyklů a to bez toho abychom vůbec šáhli na nějaké programování.

RootOf

Nyní se podíváme na funkci RootOf. Tato funkce souvisí sice s Algebrou - vrací kořeny rovnic, ale sama se nám často objevuje na výstupu z mnoha dalších příkazů Maplu. Často se nám objeví něco jako:

```
> RootOf(x^2+1=0);
```

$$\mathrm{RootOf}((\_Z\,.\,(5\,..\,10))^2+1)$$

Nemusíme se hned lekat. Výsledky jsou prostě příslušné kořeny rovnice $x^2+1=0$. Ty si můžeme vyčíslit třeba pomocí evalf nebo solve

```
> evalf(RootOf(x^2+1=0));
> solve(x^2+1=0);
```

$$\mathrm{RootOf}((\_Z\,.\,(5\,..\,10))^2+1)$$
$$\mathrm{RootOf}((\_Z\,.\,(5\,..\,10))^2+1)$$

Bohužel aplikací evalf, některé kořeny ztrácíme. Tento problém naštěstí úplně vrací funkce allvalues:

```
> allvalues(RootOf(x^4+x^2+1));
```

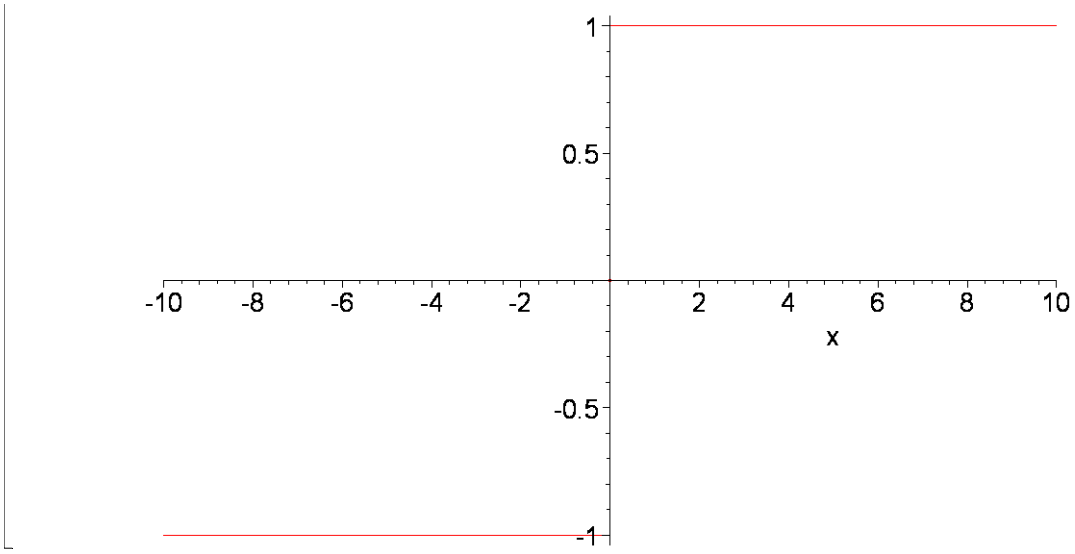
$$\mathrm{RootOf}((\_Z\,.\,(5\,..\,10))^4+(\_Z\,.\,(5\,..\,10))^2+1)$$

Funkce signum a Heaviside

Česky je to signum anglicky také, dělá to tohle:

```
> sgn(-1) = signum(-1);
> sgn( 1) = signum( 1);
> sgn( 0) = signum( 0);
> plot(signum(x), color=red,discont=true);
```

$$\mathrm{sgn}(-1)=-1$$
$$\mathrm{sgn}(1)=1$$
$$\mathrm{sgn}(0)=0$$



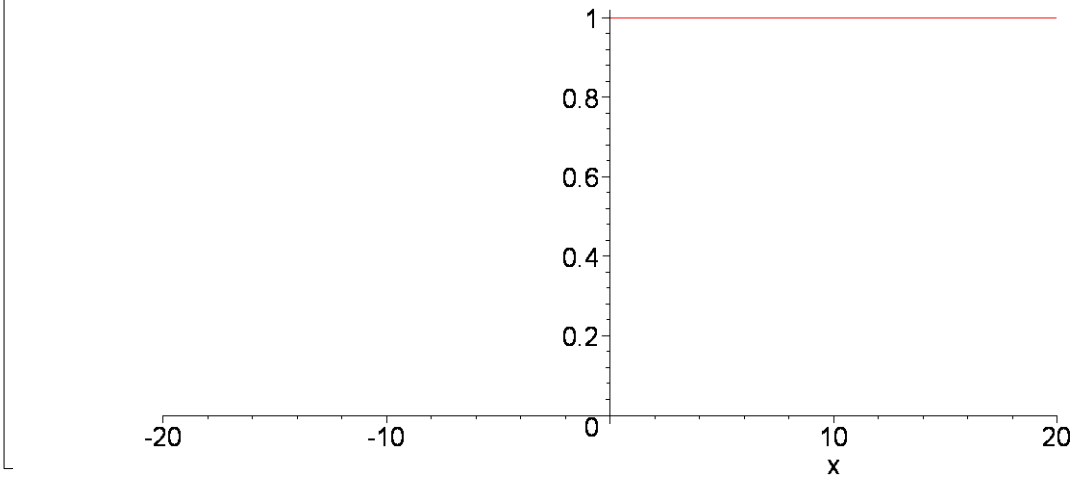
Podobně funguje funkce Heaviside:

```
> Heaviside(-1);
> Heaviside( 1);
> Heaviside( 0);
> plot(Heaviside(x),x=-20..20,color=red);
```

0

1

undefined



Tedy platí (až na x = 0) $\text{Heaviside}(x) = \frac{\text{signum}(x)+1}{2}$.

Přiřazujeme proměnným aneb assign

V některých příkladech je potřeba proměnné přiřadit nějakou hodnotu. To se dělá (kromě :=) také funkcí assign:

```
> assign(k=27);
```

```
> simplify(k^(1/3));
```

$$3$$

Pokud se předchozího přiřazení chceme zbavit napíšeme:

```
> k := 'k';
```

$$k := k$$

Totéž můžeme (globálně a brutálně) provést i pomocí restart:

```
> restart;
```

Výroba nových funkcí ze starých

Při většině problémů, které chceme v Maple spočítat, nepotřebujeme definovat vlastní funkce. Prostě rovnou napíšeme příslušnou funkcí například do integrálu apod.. Ovšem zvláště pokud kreslíme grafy a počítáme vícekrát funkční hodnoty, vyplatí se nám funkci nadefinovat. Používáme na to funkční operátor -> .

```
> f:= x -> sin(x)*exp(x^2)+cos(x)*exp(-x^2);
```

$$f := x \to \sin(x)\, \mathbf{e}^{(x^2)} + \cos(x)\, \mathbf{e}^{(-x^2)}$$

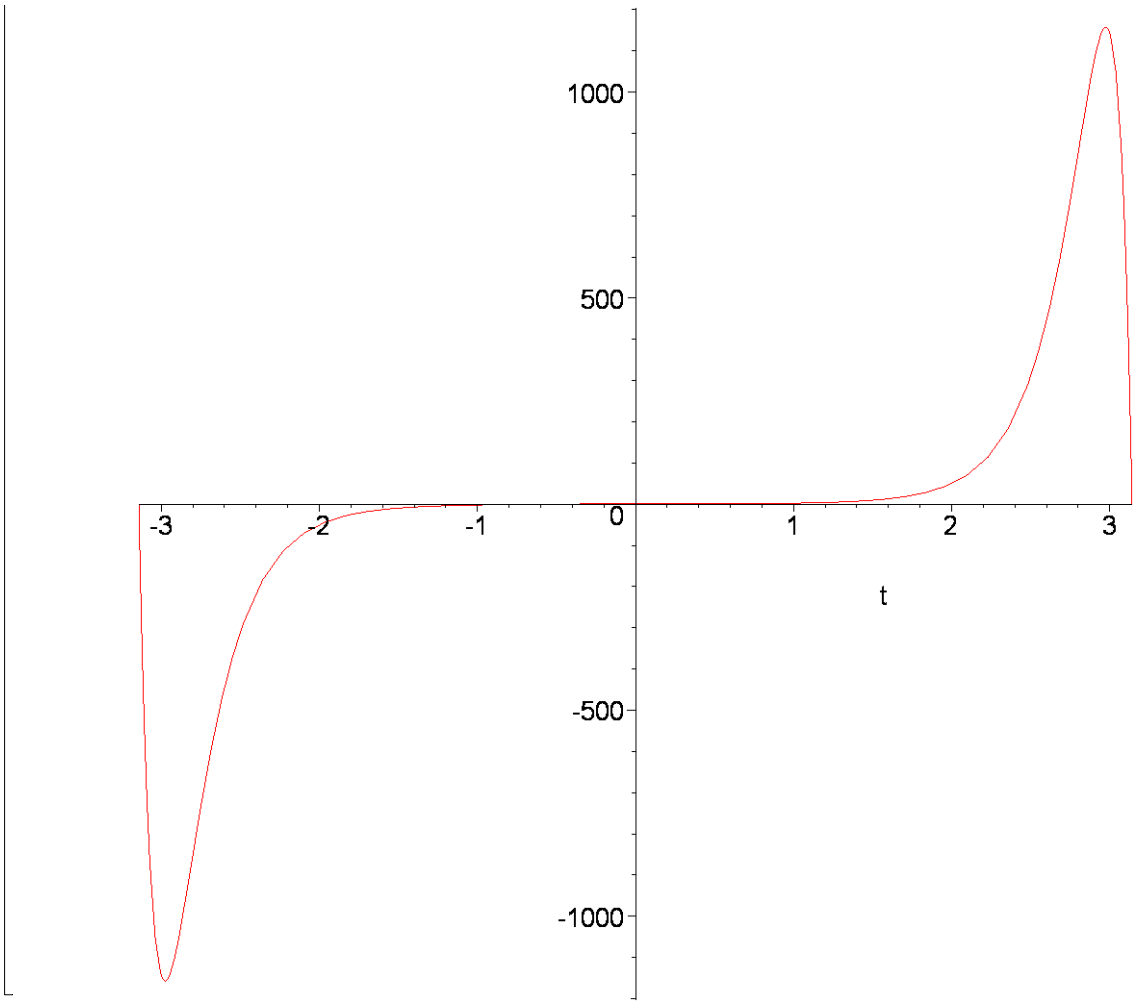
Pro získání funkční hodnoty nám pak stačí psát pouze :

```
> f(Pi);
```

$$-\mathbf{e}^{(-\pi^2)}$$

Pak můžeme snadno kreslit grafy :

```
> plot(f(t), t=-Pi..Pi);
```



Proč je důležité používat funkcionální operátor? Podíváme se co by se stalo, kdybychom funkci špatně nadefinovali :

```
> g:= sin(x)*x^3+3*x;
```

$$g := \sin(x)\, x^3 + 3\, x$$

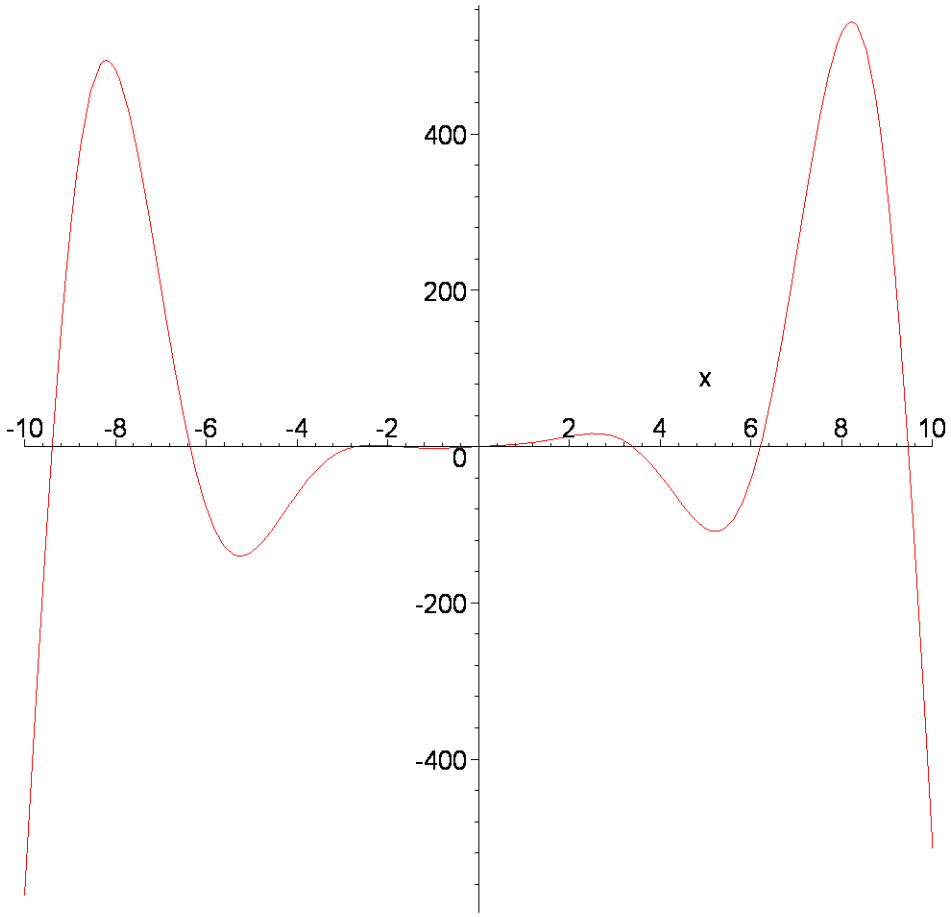
Dostáváme dost mizerné výsledky :

```
> g(Pi);
```

$$\sin(x)(\pi)\, x(\pi)^3 + 3\, x(\pi)$$

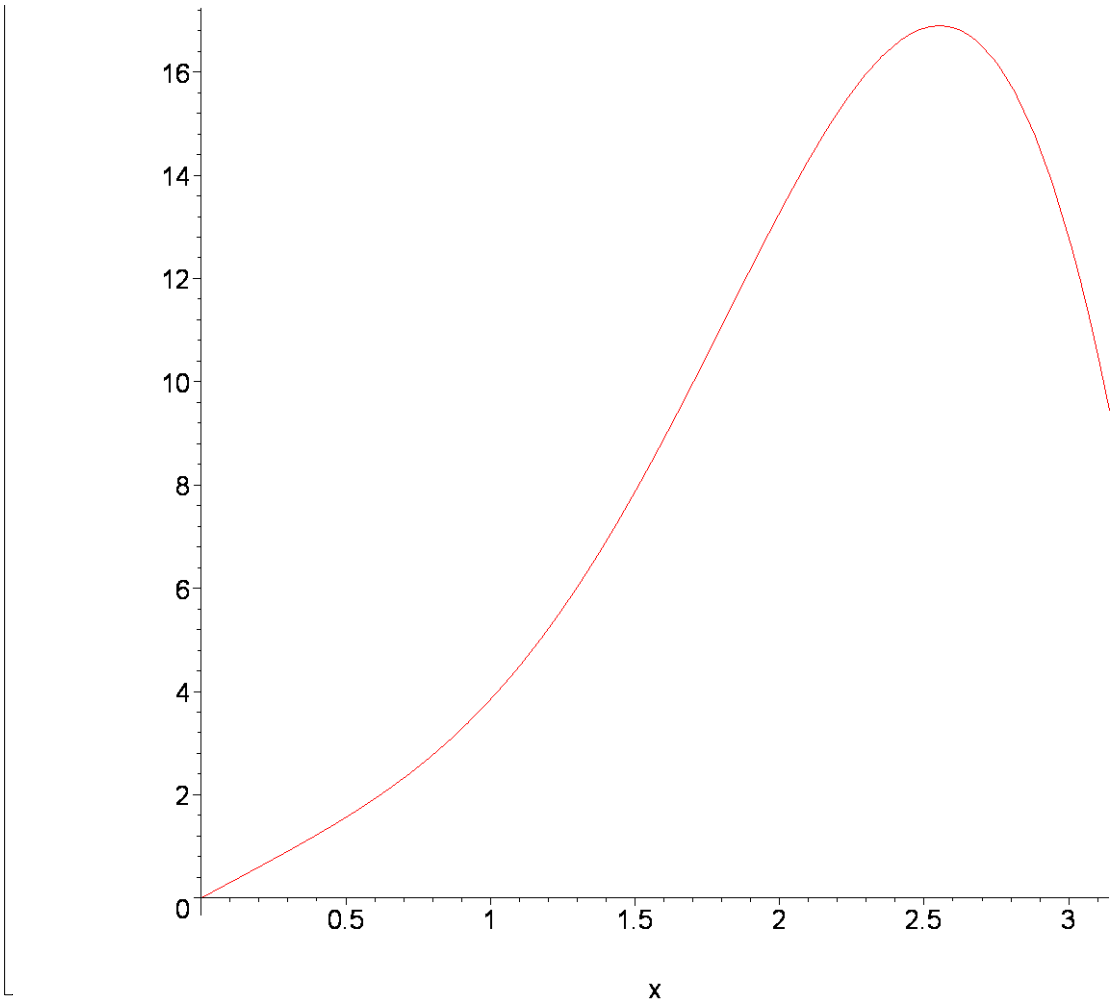
A graf taky nic moc :(:

```
> plot(g(x));
```



I když se nám podaří nakreslit graf následujícím způsobem. Doporučuji používat vždy funkční operátor

```
> plot(g, x=0..Pi);
```



Funkce můžeme definovat i po částech. Funguje to takhle:

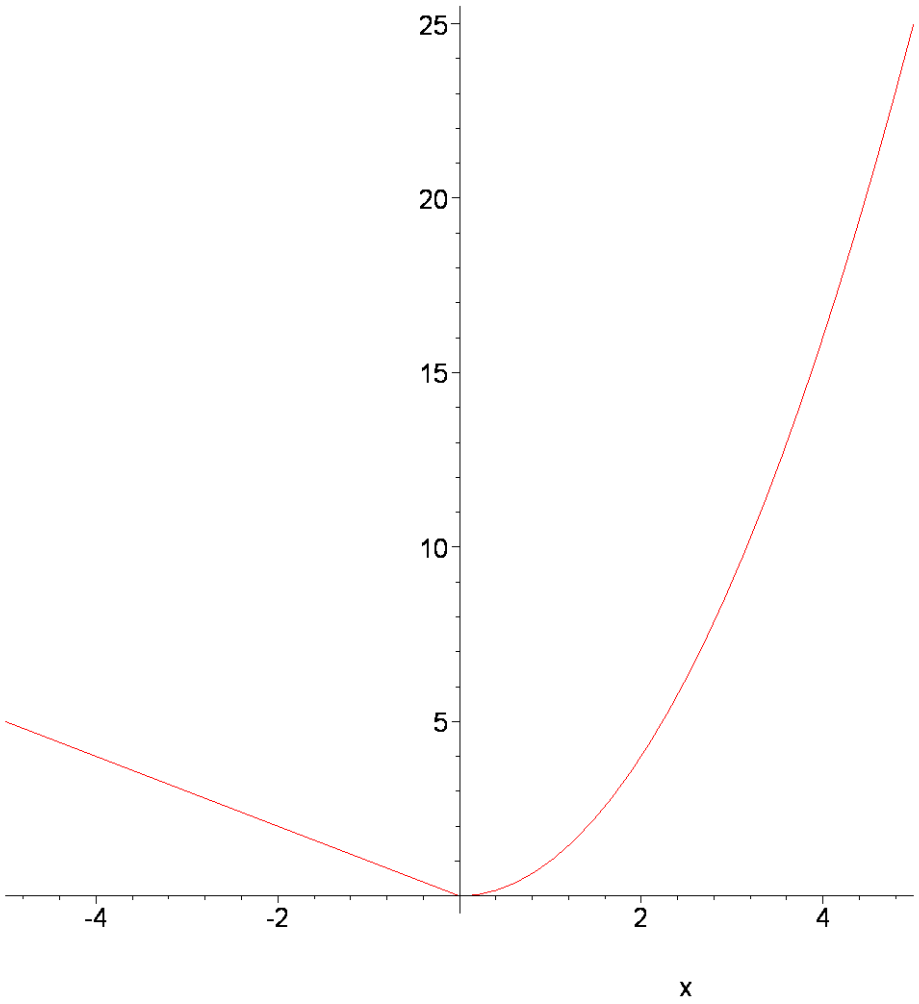
```
> piecewise(x>0,x^2,x<0, abs(x));
```

$$\begin{cases} x^2 & 0 < x \\ |x| & x < 0 \end{cases}$$

Teď si to nakreslíme:

```
> f := x -> piecewise(x>0,x^2,x<0, abs(x));
> plot(f(x),x=-5..5);
```

$$f := x \rightarrow \text{piecewise}(0 < x, x^2, x < 0, |x|)$$



Tímto způsobem můžeme samozřejmě popsat fci i na více než na dvou intervalech.

Grafy

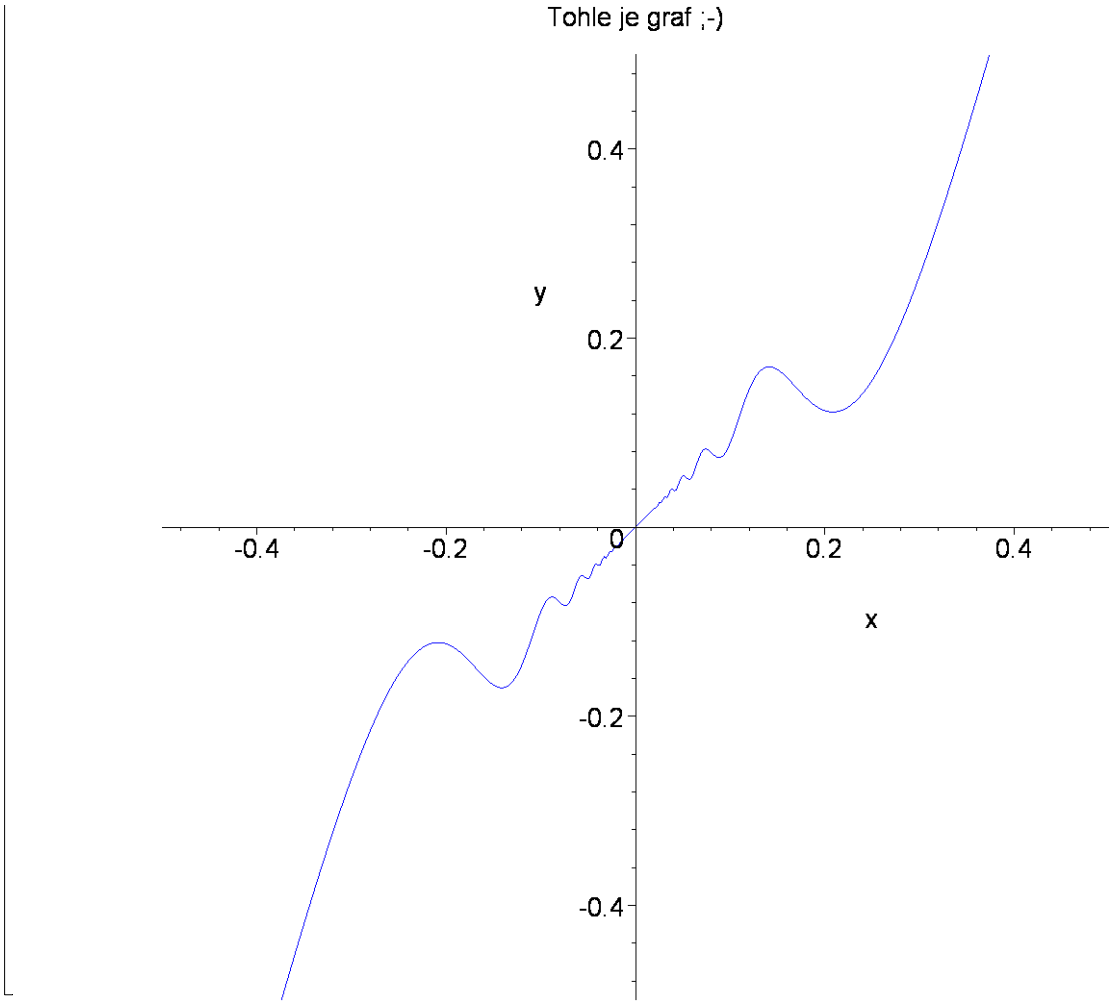
Nejdříve si zadefinujeme nějakou pěknou funkci ;-) :

```
> f := x -> x + 2*x^2*sin(1/x);
```

$$f := x \to x + 2\,x^2 \sin\left(\frac{1}{x}\right)$$

Podíváme se na nějaký její jednoduchý graf.

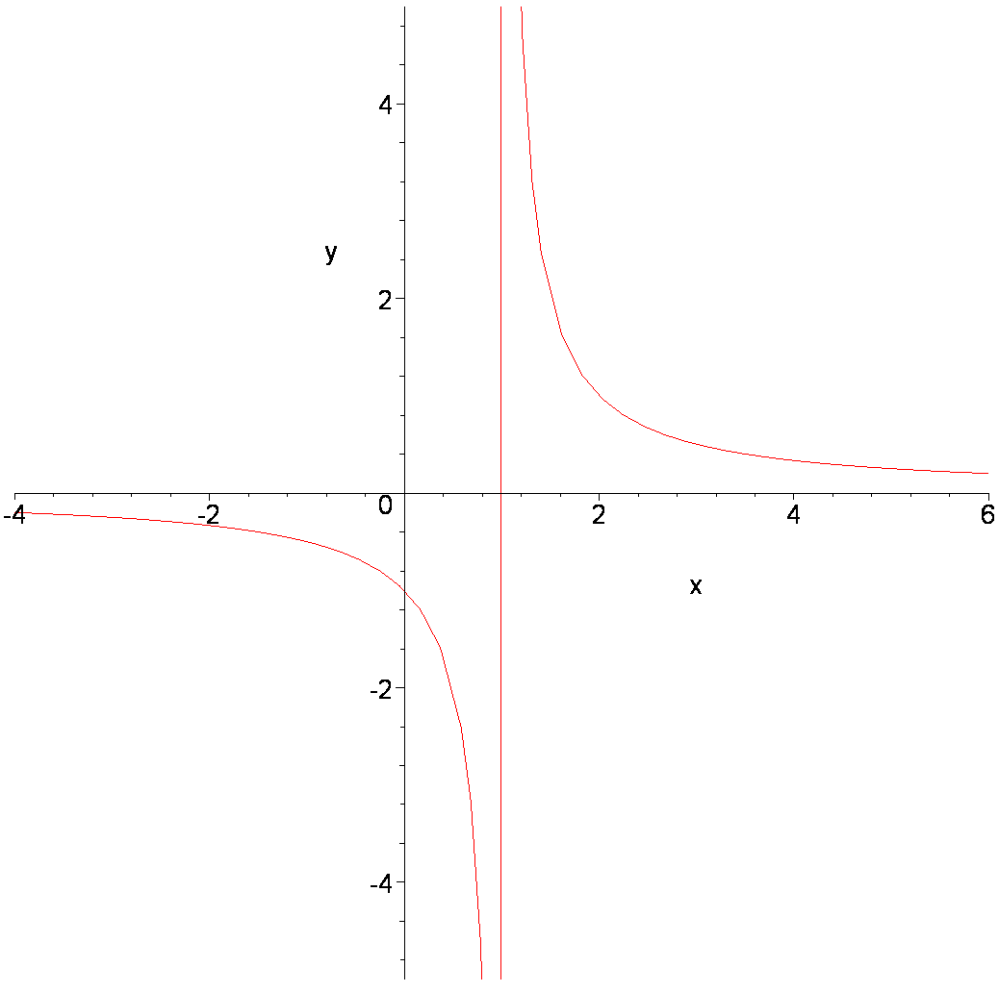
```
> plot(f(x), x=-1/2..1/2, y=-1/2..1/2, color=blue,
  numpoints=500,title=`Tohle je graf ;-)`);
```



Tak takhle pěkně jsme to nakreslily, volba numpoints určuje v kolika bodech se funkční hodnoty kreslí. Bez ní by tento graf vypadal trochu nepěkně - nepřesně a hranatě ;-), ale uvětšiny ostatních grafů není potřeba. Označení y je standartní označení pro druhou osu, toto lze předefinovat pomocí volby axes - viz help na plot a na plot[options] .

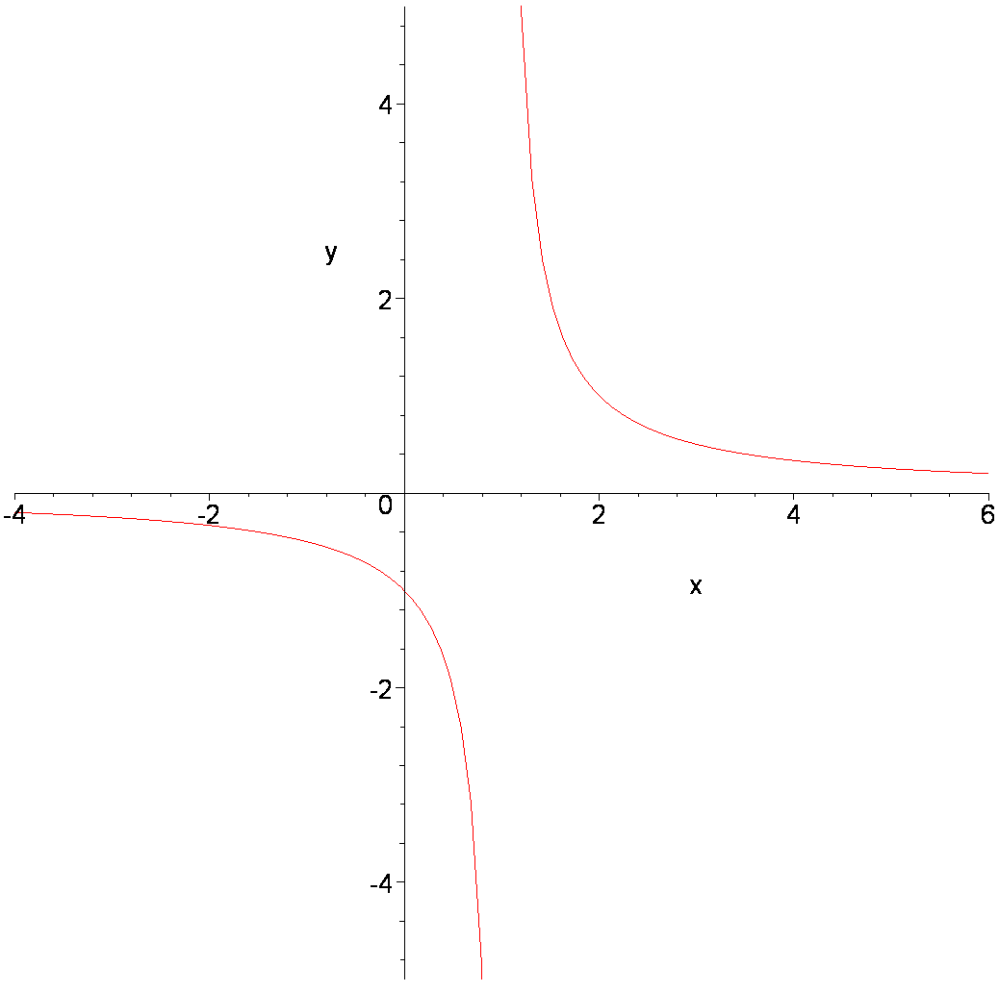
V případě, že funkce je nespojitá může nám nastat následující problém.

```
> plot(1/(x-1), x=-4..6, y=-5..5);
```



Všimněte si nehezké čáry v bodě nespojitosti x = 1. Tu odstraníme přidáním podmínky discont=true

```
> plot(1/(x-1), x=-4..6, y=-5..5, discont=true);
```



Nakreslit jeden graf nám často nestačí, a tak se teď budeme zabývat vytvářením více grafů najednou. S grafy manipuluje knihovna plots.

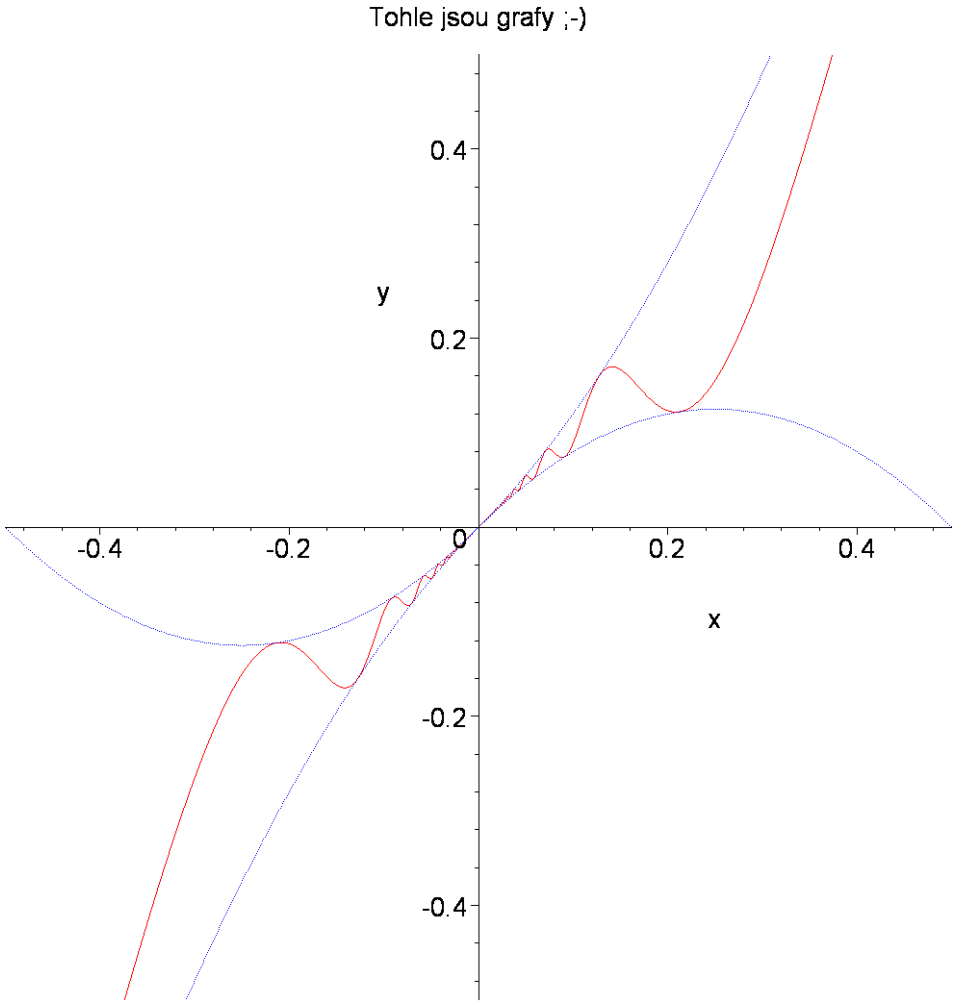
```
> with(plots);
```

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

Teď slíbených více grafů najednou. Zobrazujeme je pomocí funkce display.

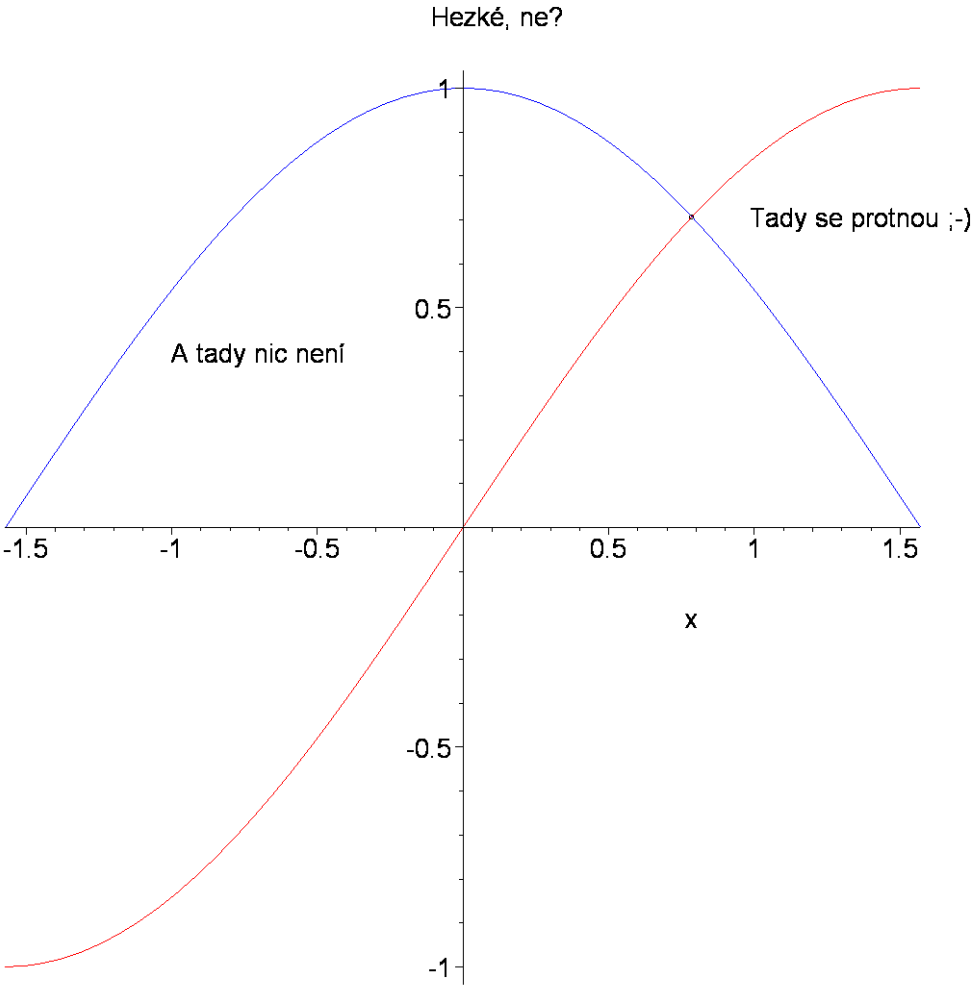
```
> plotf := plot(f(x), x=-1/2..1/2, y=-1/2..1/2, color=red,
  numpoints=500):
> plot1 := plot(x+2*x^2, x=-1/2..1/2, y=-1/2..1/2,
```

```
   linestyle=10,color=blue):
> plot2 := plot(x-2*x^2, x=-1/2..1/2, y=-1/2..1/2,
   linestyle=10,color=blue):
> display([plot1,plot2,plotf], title=`Tohle jsou grafy ;-)`);
```



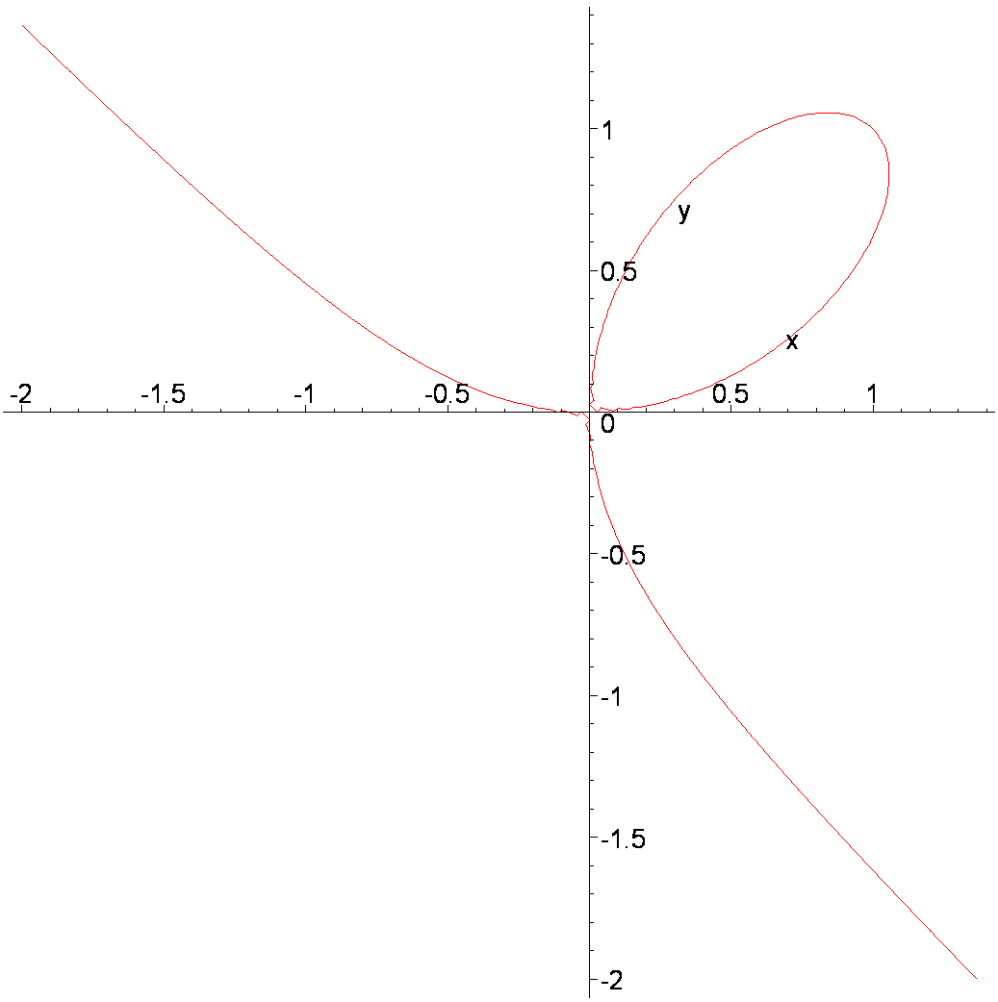
Maple umí i jiné zajímavé "grafy".

```
> plot1:= plot(sin(x),x=-Pi/2..Pi/2):
> plot2:= plot(cos(x),x=-Pi/2..Pi/2,color=blue):
> plot3:= textplot([[Pi/4+0.2,sqrt(2)/2,`Tady se protnou
   ;-)`],[-1,0.4,`A tady nic není`]],align={RIGHT}):
> plot4:= pointplot([Pi/4,sqrt(2)/2],symbol=CIRCLE):
> display({plot1,plot2,plot3,plot4}, title=`Hezké, ne?`);
```



Teď se podíváme na grafy implicitně zadané funkce. Takhle zobrazí Maple Descartesův list:

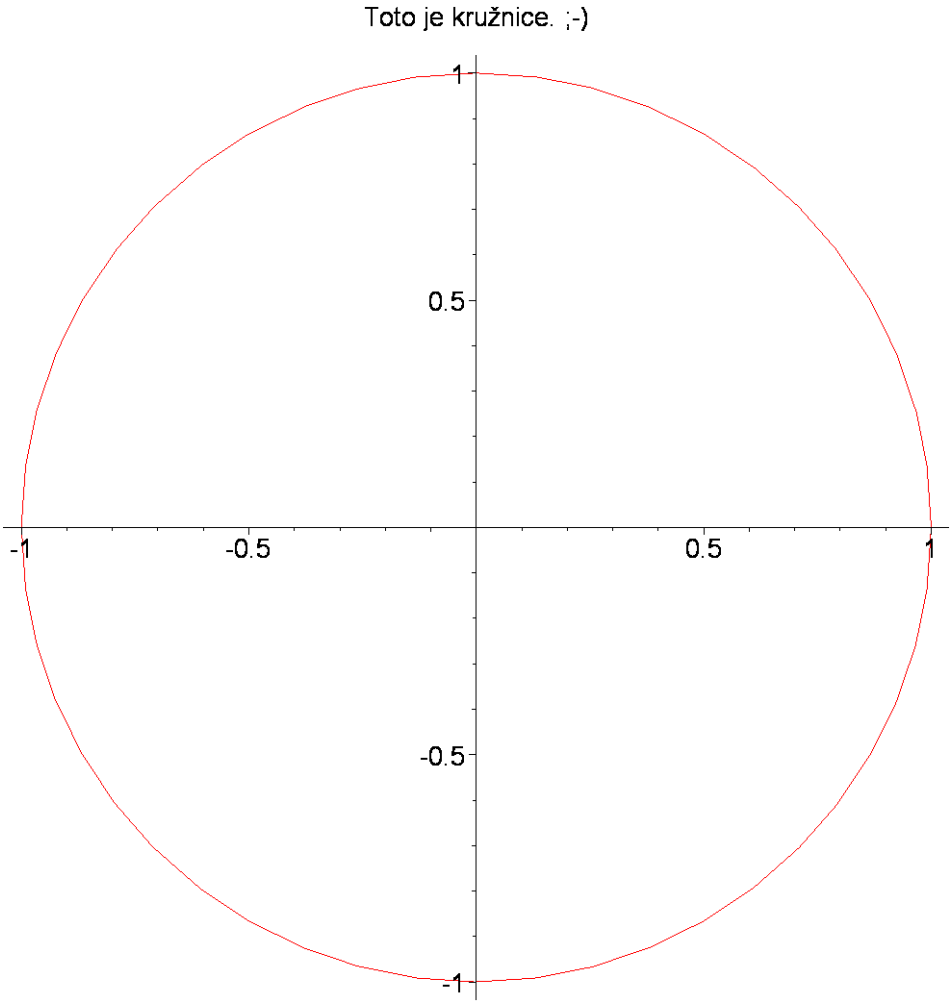
```
> implicitplot(x^3+y^3-2*x*y=0, x=-2..2,
  y=-2..2,numpoints=5000, color=red);
```



Malá poznámka k volbě numpoints: tato volba není u většiny příkladů vůbec nutná. Obyčejně stačí základní nastavení (to je 50). Toto nastavení sice často nestačilo, ale Maple je nadán "přirozenou inteligencí" a sám volí hodnoty tak aby nakreslená křivka, co nejvíce odpovídala skutečné funkci. V některých příkladech bohužel jeho "chytrost" zklame. Descartův list je jedním z nich..

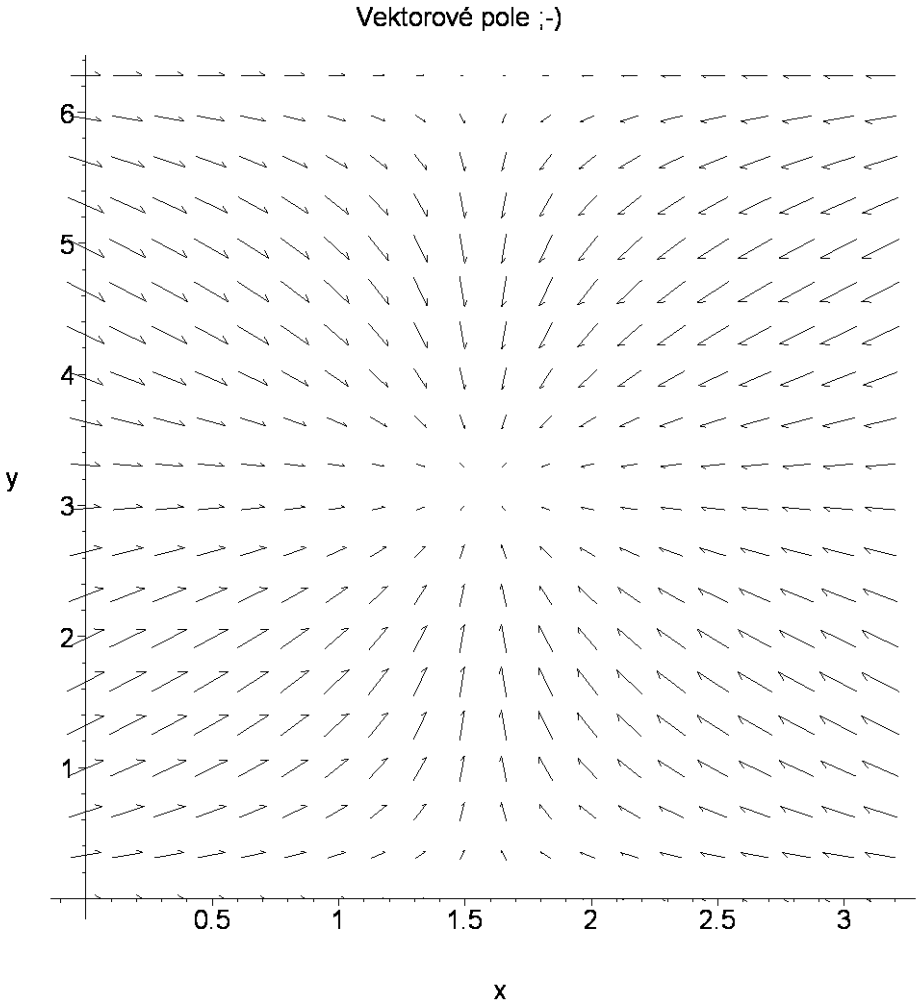
Můžeme zobrazit i paramatetrické funkce. Na to nám vystačí pouhá funkce plot.

```
> plot([cos(t),sin(t), t=0..2*Pi], title=`Toto je kružnice.
  ;-)`);
```



Maple umí ještě i další triky s dvourozměrnými grafy. Než přejdeme ke grafům v trojrozměrném prostoru ukážeme si ještě jeden kousek pro fyziky:

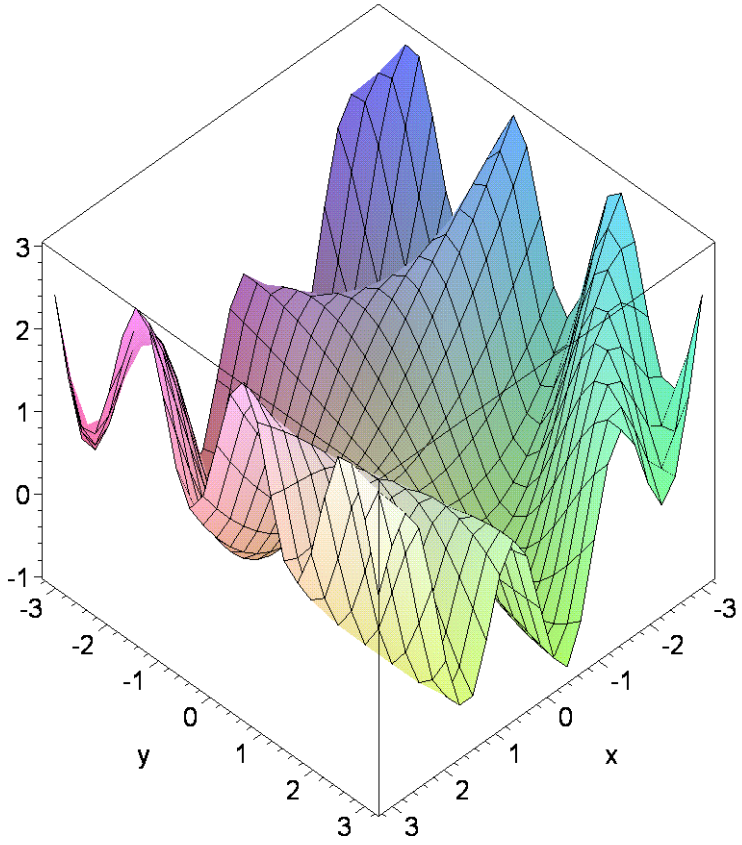
```
> fieldplot([cos(x),sin(y)],x=0..Pi,y=0..2*Pi, title=`Vektorové
  pole ;-)`);
```



V trojrozměrném prostoru kreslíme grafy pomocí plot3d:

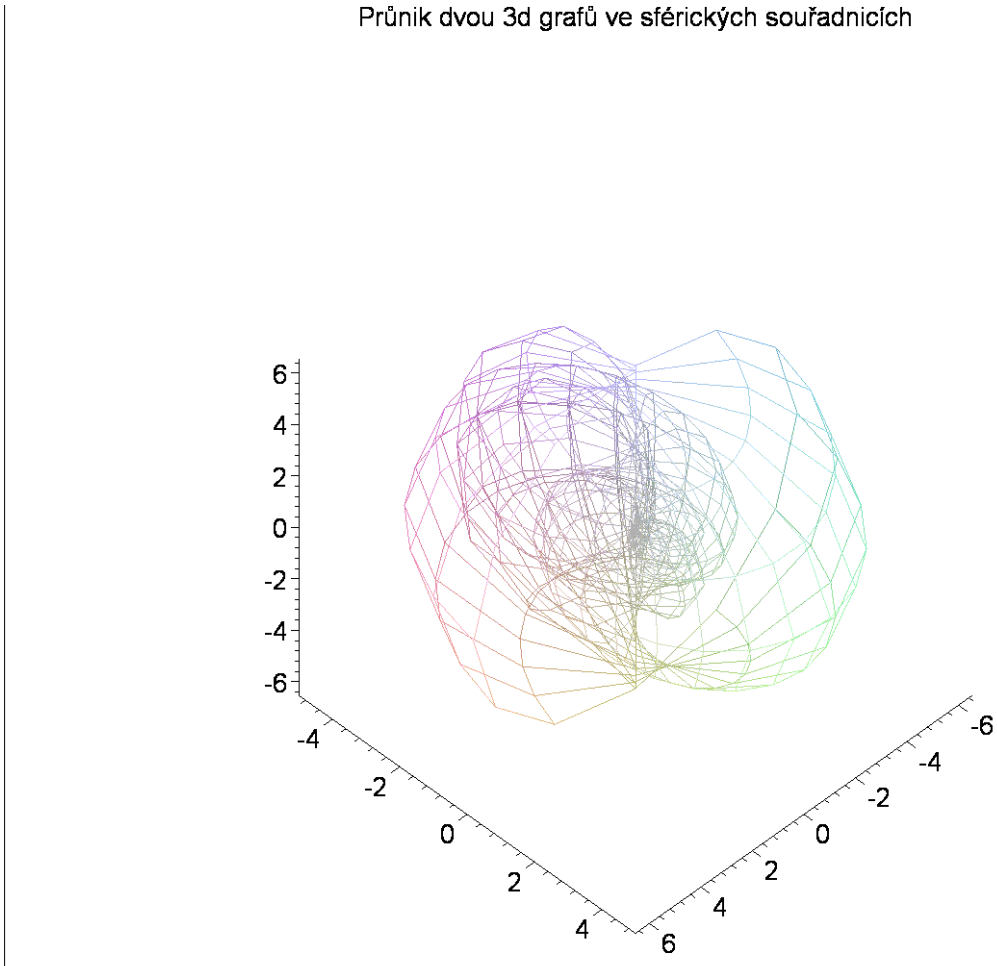
```
> plot3d(1/5*x^2+sin(x*y), x=-Pi..Pi, y=-Pi..Pi, style=patch,
  axes=boxed, title=`Takhle vypadá 3d graf ;-)`);
```

Takhle vypadá 3d graf ;-)



Grafy můžeme kreslit i v jiných souřadnicích než kartézských, například ve sférických.

```
> plot3d({y*sin(x),x}, x=0..2*Pi, y=-2*Pi..2*Pi,
  style=wireframe, coords=spherical, axes=frame, title=`Průnik
  dvou 3d grafů ve sférických souřadnicích`);
```



>

Poznámka : v nadpisu předchozího grafu můžeme vidět nemilé překvapení, které nám Maple připravuje s některými českými znaky :(. Nezbývá nám nic jiného než se těmto znakům vyhnout nebo počkat na novější verzi.

Při kreslení dalších druhů grafů Vám dopomáhej Bůh a help na plots ;-) .

To jak se kreslí grafy k numerickým řešením diferenciálních rovnic najdete v listu Základy numerické matematiky .

Použitá literatura

Vojtěch Jarník : Úvod do počtu diferenciálního (Diferenciální počet I)
Jiří Kopáček a kol.: Příklady z matematiky pro fyziky II.