Matematická analýza podporovaná wxMaxima

Rudolf Blaško

1 Úvod do wxMaxima

wxMaxima je dialogové rozhraní pro systém počítačové algebry Maxima. wxMaxima nabízí menu a dialogová okna pro běžné příkazy, automatické dokončování, vložené grafy a jednoduché animace. wxMaxima je distribuován pod licencí GPL. Maxima patří mezi Open Source programy s otevřeným zdrojovým kódem. Program je možné kompilovat v různých OS, včetně Windows, GNU/Linuxu a MacOS X. Předkompilovanými program pro GNU/Linux a Windows lze bezplatně získat na stránce SourceForge `https://sourceforge.net/projects/maxima/files/`. Po spuštění prostředí wxMaxima se na obrazovce objeví okno s menu v horní části. Pod menu se nachází prostor, kde můžeme zadávat příkazy a kde se objevují výstupy.

```
(%i1)  First input line.
(%o1)  First output line.
(%i2)  Second input line.
(%o2)  Second output line.
```

Příkazy zadáváme na samostatné řádky (vstupní řádky), jejich provedení se zajistí současným stiskem kláves `Shift` a `Enter` nebo kliknutím v menu na ikonu (Send the current cell to maxima). Vstupní řádky jsou uvedeny oznámením (%i1) a výstupní řádky jsou uvedeny oznámením (%o1). Čísla pro vstupní a k němu odpovídající výstupní řádek jsou identické a na základě tohoto čísla se můžeme na obsah těchto řádků odvolávat.

```
(%i1)  solve(0=x+2,x);
(%o1)  [x = -2]
(%i2)  %i1;
(%o2)  solve(0 = x + 2, x)
(%i3)  %o1;
(%o3)  [x = -2]
```


DOI: https://doi.org/10.11118/978-80-7509-898-6-0050

Příkazy se provedou na nové samostatmé řádky (výstupní řádky). Příkazy na vstupních řádcích můžeme ukončit symbolem `;` (který systém automaticky doplní) nebo symbolem `$`, který potlačí zobrazení příslušného výstupu. Na vstupní řádek můžeme zadat i více příkazů, ale musíme je oddělit symboly `;` nebo `$`. Příkaz můžeme také strukturovat na více vstupních řádků.

```
(%i1)  a:2;b:3;solve(a*x+b*x^2=0,x)
(a)    2
(b)    3
(%o1)  [x = -2/3, x = 0]
(%i2)  a:2$ b:3$ solve(a*x+b*x^2=0,x);
(%o2)  [x = -2/3, x = 0]
(%i3)  a:2$
       b:3$
       solve(a*x+b*x^2=0,x);
(%o3)  [x = -2/3, x = 0]
```

Výstup můžeme uložit v různých tvarech a následně použít v jiných programech (LaTeX, editor rovnic MSWord, ...). Výstup (%o3) z předchozího okna můžeme:

- kopírovat (`CRL C` a `CRL V`), resp. kopírovat jako text (lze použít např. pro editor rovnic MSWord): `x=-2/3, x=0`,
- kopírovat jako LaTeX `\[x=-\frac{2}{3}\operatorname{,}x=0\]`,
- kopírovat jako MathML, obrázek, RTF, SVG...

Prostředí wxMaxima má dobře propracovaný help pro uživatele, který najdeme v menu `Help`. Help otevřeme také stisknutím klávesy `F1`. Manuál najdeme také na webové stránce `https://maxima.sourceforge.io/docs/manual/maxima_369.html`.

1.1 Základní příkazy

Příkazem `apropos` můžeme zjistit přesný název příkazu pomocí části jeho názvu.

```
(%i1)  apropos("plot")
(%o1)  [barsplot,boxplot,contour_plot,get_plot_option,gnuplot,...
```

Příkaz `describe` vypíše popis zadaného příkazu.

```
(%i1)  describe(plot2d);
       - -Function: plot2d
          plot2d (<expr><, <range_x><, <options><)
          plot2d (<expr_<>=<expr_<>, <range_x><, <range_y><, <options><)
          ...
(%o1)  true
```

Výrazy se zadávají pomocí obvyklých znaků operací, relací a funkcí. Argumenty funkcí a příkazů jsou v kulatých závorkách, symbol násobení `*` se musí zadat! Umocnění se zadává znakem `^` nebo dvojicí `**`.

Symbol `:` slouží k přiřazení hodnoty napravo výrazu na nalevo.

```
(%i1)  a:2$ b:3$ solve(a*x+b*x^2=0,x);
(%o1)  [x = -2/3, x = 0]
```

V menu `View` a podmenu `Display Equations` můžeme změnit zobrazení výstupních řádků na tvary `in 2D`, `as 1D ASCII` nebo `as ASCII Art`. Implicitně je nastaveno `in 2D`. Nastavení výstupu můžeme změnit i příkazem `set_display`. Nastavení na tvar `in 2D` má argument `none`.

```
(%i1)  x/sqrt(x^2+1);set_display('none)$

(%o1)  x/sqrt(x^2+1)              /* in 2D */
```

Pomocí argumentu `ascii` v příkazu `set_display` změníme výstup na tvar `as 1D ASCII` a pomocí argumentu `xml` na tvar `as ASCII Art`.

```
(%i1)  x/sqrt(x^2+1);set_display('ascii)$
(%o1)  x/sqrt(x^2+1)              /* as 1D ASCII */
(%i2)  x/sqrt(x^2+1);set_display('xml)$
                 x
(%o2)  ------------               /* as ASCII Art */
                2
       sqrt(x +1)
```

Příkazem `kill` můžeme odstranit proměnné se všemi jejich parametry a vlastnostmi z paměti.

```
(%i1)  kill(a,b)          /* removes all bindings from the arguments a,b */
(%i2)  kill(all)          /* removes all items on all infolists */
```

1.2 Práce s čísly a základní konstanty

Maxima může pracovat s reálnými čísly zapsanými v numerickém nebo symbolickém tvaru. Způsob zápisu reálných čísel můžeme nastavit v menu `Numeric` pomocí přepínače `Numeric Output` mezi numerickým a symbolickým zobrazováním. Také zde můžeme zvolit způsob a presnosnosť numerického zobrazování. O způsobu zobrazování rozhoduje nastavení proměnné `numer`.

Standardně se zobrazuje 16 číslic (včetně desetinné tečky). Přesnost zobrazení definuje proměnná `fpproc` a ovlivňuje zobrazení pomocí `bfloat`. Výstup `float` zobrazuje vždy stejně. Přesnost můžeme prakticky neomezeně zvýšit nebo snížit. Můžeme ji změnit globálně a také lokálně pouze pro jednu přeměnu nebo příkaz.

```
(%i1)  log(2);
(%o1)  log(2)
(%i2)  log(2),numer;
(%o2)  0.6931471805599453
(%i3)  float(log(2));
(%o3)  0.6931471805599453
(%i4)  bfloat(log(2));
(%o4)  6.931471805599453b-1
(%i5)  log(2),bfloat;
(%o5)  6.931471805599453b-1
(%i6)  bfloat(log(2)),fpprec=34;
(%o6)  6.9314718055994530941723212145817666b-1
(%i6)  bfloat(log(2)),fpprec=134;
(%o6)  6.9314718055994530941723212145[78digits]10205706857336855202357581 3b-1
```

Číselné konstanty e, π, i (imaginární jednotka) mají prefix `%`, tj. `%e`, `%pi`,`%i`. To platí i pro konstanty, které jsou součástí nebo výsledkem výpočtů. Také mají prefix `%`.

Maxima má předdefinované konstanty `inf`, `minf` pro reálné nekonečná ∞, $-\infty$ a `infinity` pro komplexní nekonečno.

Logické konstanty `true` a `false` představují pravdu a nepravdu.

```
(%i1)  %pi; %i; %e;
(%o1)  π %i %e
(%i2)  minf; inf;
(%o2)  −∞ ∞
(%i3)  infinity;
(%o3)  infinity
```

Komplexním číslům se v tomto kurzu nevěnujeme, proto pouze zmíníme jak se zobrazují. Komplexní čísla se implicitně zadávají algebraickém tvaru (`rectform`). Do goniometrického (exponenciálního) tvaru je můžeme převést pomocí příkazu `polarform`.

```
(%i1)  z:1+%i;
(z)    i+1
(%i2)  polarform(z)+rectform(z);
(%o2)  √2 e^(i π/4) +i+1
```

1.3 Přiřazení a funkce

Operátor `:` používáme na přiřazení hodnot nebo výrazů proměnným. Funkce definujeme pomocí přiřazení `:=`.

```
(%i1)  f(x):=x^2+2*x+3;
(%o1)  f(x) := x^2 + 2 * x + 3
(%i6)  f(x);f(y);f(x+1);f(-2);f(1);
(%o2)  x^2 + 2 * x + 3
(%o3)  y^2 + 2 * y + 3
(%o4)  (x + 1)^2 + 2 * (x + 1) + 3
(%o5)  3
(%o6)  6
```

Maxima obsahuje mnohem více funkcí než standardní programovací jazyky. Jsou to nejen samotné reálné funkce, ale také různé funkce pro jejich podporu. Mezi základní funkce patří `sign(x)`, `abs(x)`, `floor(x)` (dolní celá část čísla x), `round(x)` (zaokrouhlí x na nejbližší celé číslo), `truncate(x)` (odstraní všechny číslice za desetinnou tečkou), `ceiling(x)` (horní celá část čísla x).

```
(%i2)  f(x):=sign(x)$ print(f(-3.2),f(0),f(3.2))$
       neg zero pos
(%i4)  f(x):=abs(x)$ print(f(-3.2),f(0),f(3.2))$
       3.2 0 3.2
(%i6)  f(x):=floor(x)$ print(f(-3.6),f(-3.2),f(-1),f(0),f(1),f(3.2),f(3.6))$
       -4 -4 -1 0 1 3 3
(%i8)  f(x):=round(x)$ print(f(-3.6),f(-3.2),f(-1),f(0),f(1),f(3.2),f(3.6))$
       -4 -3 -1 0 1 3 4
(%i10) f(x):=truncate(x)$ print(f(-3.6),f(-3.2),f(-1),f(0),f(1),f(3.2),f(3.6))$
       -3 -3 -1 0 1 3 3
(%i12) f(x):=ceiling(x)$ print(f(-3.6),f(-3.2),f(-1),f(0),f(1),f(3.2),f(3.6))$
       -3 -3 -1 0 1 4 4
```

Pro formátování výpisu jsme použili příkaz `print`.

```
(%i3)  a:2$ b:log(2),numer$ print("Logarithm of a-number",a," is ",log(a),"=",b)$
       Logarithm of a number 2 is log (2) = 0.6931471805599453
```

Maxima obsahuje mnoho elementárních funkcí. Jsou to například `exp(x)=%e^x`, `log(x)`, goniometrické funkce a k nim inverzní funkce `sin(x)`, `cos(x)`, `tan(x)`, `cot(x)`, `asin(x)`, `acos(x)`, `atan(x)`, `acot(x)`, hyperbolické a k nim inverzní funkce `sinh(x)`, `cosh(x)`, `tanh(x)`, `coth(x)` `asinh(x)`, `acosh(x)`, `atanh(x)`, `acoth(x)` atd.

Maxima obsahuje také mnoho funkcí pro jejich podporu. Některé z nich nejsou implementovány přímo v prostředí wxMaxima, ale v externích knihovnách nazývaných balíčky (packages). Tyto balíčky se do systému načítají pomocí příkazu `load`. Na ukázku uvedeme balíček `spangle` pro podporu práce s goniometrickými funkcemi.

```
(%i2)  print(tan(%pi/8),ratsimp(tan(%pi/8)),trigsimp(tan(%pi/8)))$

       tan(π/8)  tan(π/8)  sin(π/8)/cos(π/8)

(%i3)  load(spangl);

(%o3)  ../share/trigonometry/spangl.mac

(%i4)  tan(%pi/8);

(%o4)  √2 − 1
```

1.4 Práce s výrazy

Operace a výpočty programu Maxima probíhají v nějakém prostředí, ve kterém systém předpokládá platnost určitých podmínek. Tyto podmínky můžeme měnit. Mnohokrát potřebujeme změnit podmínky pouze lokálně pro nějaký konkrétní výpočet aniž

abychom měnili globální nastavení. K tomuto účelu posytuje Maxima velmi účinný příkaz `ev`, který umožňuje definovat specifické prostředí v rámci jednoho příkazu.

Po zadání příkazu `ev(a,b1,b2,...,bn)` se vyhodnotí výraz `a` při splnění podmínek `b1`, `b2`, ..., `bn`. Tyto podmínky mohou být rovnice, přiřazení, funkce, přepínače (logické nastavení). Na ukázku uvádíme příklad řešení kvadratické rovnice pomocí příkazu `solve`. Proměnné `a`, `b`, `c` po provedení příkazu `ev` nemají přiřazené hodnoty.

```
(%i1)  ev(solve(a*x^2+b*x+c=0,x),a:2,b:-1,c=-3);
(%o1)  [x = 3/2, x = -1]
(%i2)  solve(a*x^2+b*x+c=0,x);
(%o2)  [x = -(sqrt(b^2-4*a*c)+b)/(2*a), x = (sqrt(b^2-4*a*c)-b)/(2*a)]
```

Na zjednodušení a úpravy různých výrazů nabízí Maxima několik příkazů. Základní funkce najdeme v menu `Simplify`. S příkazy `ratsimp` a `trigsimp` jsme se již setkali a při úpravě hodnoty `tan(%pi/8)` neměli žádoucí efekt.

Maxima nabízí pomocí příkazu `example` příklady k jednotlivým příkazům. Podívejme se na některé ukázky, které nabízí `example(ratsimp)`.

```
(%i2)  f(x):=b*(a/b-x)+b*x+a$ print(f(x),"?",ratsimp(f(x)))$
       bx + b(a/b - x) + a ? 2a
(%i3)  ratsimp(a+1/a);
(%o3)  (a^2+1)/a
(%i4)  ev(x^(a+1/a),ratsimp);
(%o4)  x^(a+1/a)
(%i5)  ev(x^(a+1/a),ratsimpexpons);
(%o5)  x^((a^2+1)/a)
```

Funkce `expand` roznásobí příslušné členy ve výrazu. Funkce `factor` daný výraz naopak rozloží. Funkce `gfactor` tak činí nad polem komplexních čísel.

```
(%i1)  f(x):=(x+1)*(x^2-4)*(x^2+4)$
(%i3)  ratsimp(f(x));expand(f(x));
(%o2)  x^5 + x^4 - 16*x - 16
(%o3)  x^5 + x^4 - 16*x - 16
(%i6)  factor(f(x));gfactor(f(x));factor(100);
```

(%o4) $(x-2)(x+1)(x+2)(x^2+4)$
(%o5) $(x-2)(x+1)(x+2)(x-2\%\mathrm{i})(x+2\%\mathrm{i})$
(%o6) $2^2 5^2$

Racionální lomenou funkci rozložíme na parciální zlomky pomocí příkazu `partfrac`.

```
(%i1) partfrac((x+1)/(x^2-2*x+1),x);
```
(%o1) $\frac{1}{x-1}+\frac{2}{(x-1)^2}$

Substituovat výrazy můžeme pomocí příkazů `subst(a,b,c)` a `ratsubst(a,b,c)`. Výraz `a` bude nahrazen za výraz `b` a následně dosazen do výrazu `c`. Při použití příkazu `subst` musí být `b` nejjednodušší částí (atomem) nebo kompletním podvýrazom výrazu `c`. V příkladu není podvýraz `x+y` kompletní (chybí `z`). Příkaz `ratsubst` výsledný výraz i upraví.

```
(%i2) subst(x+y,a,a^2+b^2);ratsubst(x+y,a,a^2+b^2);
```
(%o1) $(y+x)^2+b^2$
(%o2) $y^2+2xy+x^2+b^2$
```
(%i4) subst(a,x+y,x+y+z);ratsubst(a,x+y,x+y+z);
```
(%o3) $z+y+x$
(%o4) $z+a$

1.5 Limity a derivování

V menu `Calulus` najdeme funkce na řešení základních úloh matematické analýzy (limity, derivování, integrování, součty řad, rozklad funkce do Taylorova polynomu...).

Limity počítáme pomocí příkazu `limit`. Poslední parametr určuje směr jednostranných limit, má hodnoty `plus`, resp. `minus` a je nepovinný. Pokud není určen, Maxima počítá limitu jako komplexní. Příkazy `limit(f(x),x,a)`, `limit(f(x),x,a,plus)` vypočítáme limity $\lim_{x\to a} f(x)$, $\lim_{x\to a^+} f(x)$.

```
(%i4) limit(1/x,x,0);limit(1/x,x,0,plus);limit(1/x,x,0,minus);limit(1/x,t,0);
```
(%o1) $infinity$
(%o2) ∞
(%o3) $-\infty$
(%o4) $\frac{1}{x}$

Pokud použijeme před příkazem apostrof `'`, příkaz se neprovede, pouze zobrazí.

```
(%i2)  limit(((1-n)/(1+3*n))^(1+4*n),n,inf);'limit(((1-n)/(1+3*n))^(1+4*n),n,inf);
(%o1)  0
(%o2)  lim_{n→∞} ((1-n)/(3n+1))^(4n+1)
```

Derivace počítáme pomocí příkazu `diff`. Parametr, který určuje rád derivace je nepovinný.

```
(%i4)  f(x):=2*x^4-3*x+sin(x);
       print("f'=",diff(f(x),x),"=",diff(f(x),x,1))$
       print("f''=",diff(diff(f(x),x),x),"=",diff(f(x),x,2),"=",diff(f(x),x,1,x,1))$
       print("f^(10)=",diff(f(x),x,10),"=",diff(f(x),x,1,x,9))$
(%o1)  f(x) := 2x^4 - 3x + sin(x)
       f' = cos(x) + 8x^3 - 3 = cos(x) + 8x^3 - 3
       f'' = 24x^2 - sin(x) = 24x^2 - sin(x) = 24x^2 - sin(x)
       f_(10) = - sin(x) = - sin(x)
```

Parciální derivace počítáme pomocí stejného příkazu.

```
(%i3)  g(x,y):=x^3*y^2-1;
       print("g'_x=",diff(g(x,y),x),", resp. g'_y=",diff(g(x,y),y,1))$
       print("g''_(xx)=",diff(g(x,y),x,2),", resp. g''_(yx)=",diff(g(x,y),y,1,x,1))$
(%o1)  g(x,y):=x^3*y^2-1
       g'_x = 3x^2 y^2, resp. g'_y = 2x^3 y
       g''_(xx) = 6xy^2, resp. g''_(yx) = 6x^2 y
```

Taylorův polynom n-tého stupně vypočítáme pomocí příkazu `taylor`. Tento příkaz najdeme v menu `Calculus` a podmenu`Get Series...`. Taylorova řada funkce f stupně n v středu c vypočítáme příkazem `taylor(f(x),x,c,n)`. Jeho koeficienty dostaneme použitím příkazu `coeff`. Použití tohoto příkazu je závislé na příkazu `taylor`.

```
(%i1)  t1:taylor(sin(x),x,0,5); t2:taylor(sin(x),x,-1,5);
(t1)   x - x^3/6 + x^5/120 + ...
(t2)   - sin(1) + cos(1)(x+1) + sin(1)(x+1)^2/2 - cos(1)(x+1)^3/6 - sin(1)(x+1)^4/24 + cos(1)(x+1)^5/120 + ...
(%i3)  print(coeff(sin(x),x,5)," and ",coeff(t1,x,5)," and ",coeff(t2,x,5))$
       0 and 1/120 and cos(1)/120
```

Taylorův polynom polynomu je opět polynom, pouze je vyjádřen v jiném tvaru. Prakticky se změní pouze souřadnicový systém, ve kterém polynom vyjadřujeme. Začátek systému se posune z bodu 0 do bodu -1. V následujícím příkladu je Taylorův polynom daného polynomu vypočítaný i jiným způsobem. Příkaz `taylor` dává na konec tři tečky, i když je rozvoj uzavřen.

```
(%i1)  f(x):=2*x^5-x^4-3*x^3-x+1;
```
(%o1) $f(x) := 2x^5 - x^4 + (-3)x^3 - x + 1$
```
(%i2)  tp1:taylor(f(x),x,-1,5);
```
(tp1) $2 + 4(x+1) - 17(x+1)^2 + 21(x+1)^3 - 11(x+1)^4 + 2(x+1)^5 + \cdots$
```
(%i4)  ratsimp(tp1);expand(tp1);
```
(%o3) $2x^5 - x^4 - 3x^3 - x + 1$

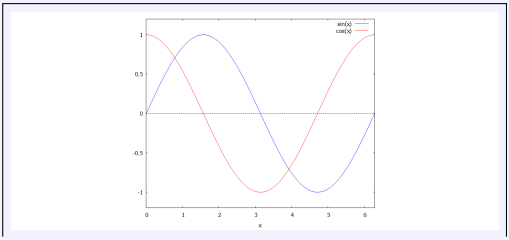
(%o4) $2x^5 - x^4 - 3x^3 - x + 1$
```
(%i6)  tpx:ratsubst(t,x+1,f(x));subst(x+1,t,tpx);
```
(tpx) $2t^5 - 11t^4 + 21t^3 - 17t^2 + 4t + 2$

(tp2) $2(x+1)^5 - 11(x+1)^4 + 21(x+1)^3 - 17(x+1)^2 + 4(x+1) + 2$
```
(%i7)  tp1-tp2;
```
(%o7) $0 + \cdots$

1.6 Grafy funkcí

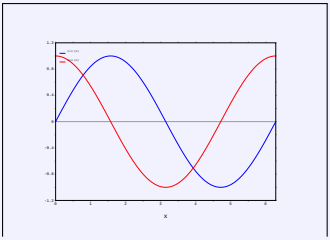
Graf funkce můžeme vykreslit několika způsoby. Nejjednodušší je zvolit v menu `Plot` podmenu `Plot 2d...`. Pokud zvolíme `Format=gnuplot`, funkci vykreslí příkaz `plot2d` pomocí Open Source programu `gnuplot` do nového okna. `gnuplot` se automaticky instaluje spolu s programem Maxima.

```
(%i1)  plot2d([sin(x),cos(x)],[x,-%pi,2*%pi],[y,-1.2,1.2],[plot_format, gnuplot])$
```



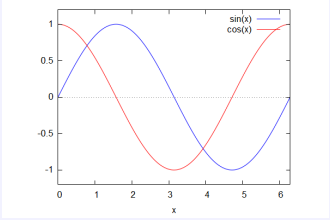
Pokud zvolíme `Format=wxmaxima`, Maxima vykreslí graf pomocí příkazu `plot2d` do nového okna. Obrázek můžeme uložit pouze do postscriptu.

```
(%i1)  plot2d([sin(x),cos(x)],[x,-%pi,2*%pi],[y,-1.2,1.2],[plot_format, xmaxima])$
```



Pokud zvolíme `Format=inline`, Maxima vykreslí graf pomocí příkazu `wxplot2d` do svého prostředí.

```
(%i1)  wxplot2d([sin(x),cos(x)],[x,-%pi,2*%pi],[y,-1.2,1.2])$
```

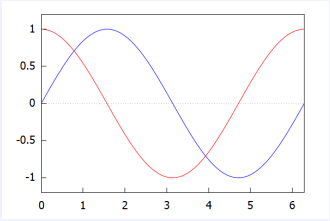


(%o1)

Příkazy `plot2d` a `wxplot2d` mají stejnou syntaxi a mají mnohem více parametrů. Parametry můžeme zjistit například příkazem `describe(plot2d)`.

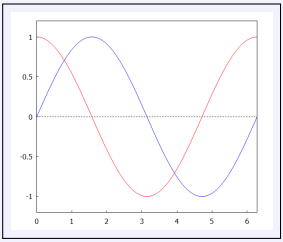
Na tisk grafu funkcí je výhodnější použít příkaz `wxdraw2d` nebo `draw2d`, který je vhodné směřovat na výstup programu `gnuplot`. Tyto příkazy mají trochu jinou syntaxi jako příkazy `wxplot2d`, resp. `plot2d`. Parametry tisku jsou v nich jednodušší a přehlednější. Vykreslována funkce musí být umístěna v příkazu `explicit`, `parametric` nebo `implicit`.

```
(%i1) wxdraw2d(xaxis=true,yaxis=true,xrange=[0,2*%pi],yrange=[-1.2,1.2],
      color=blue,explicit((sin(x)),x,0,2*%pi),
      color=red,explicit((cos(x)),x,0,2*%pi))$
```



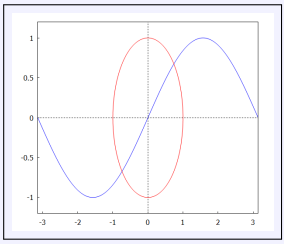
(%o1)

```
(%i1) draw2d(xaxis=true,yaxis=true,xrange=[0,2*%pi],yrange=[-1.2,1.2],
      color=blue,explicit((sin(x)),x,0,2*%pi),
      color=red,explicit((cos(x)),x,0,2*%pi))$
```



Parametrickou křivku nebo funkci vykreslíme podobným způsobem.

```
(%i1)  draw2d(xaxis=true,yaxis=true,xrange=[-%pi,%pi],yrange=[-1.2,1.2],
       color=blue,explicit((sin(x)),x,-%pi,%pi),
       color=red,nticks=300,parametric(cos(t),sin(t),t,0,2*%pi))$
```



2 Posloupnosti a řady

Posloupnosti můžeme v programu Maxima vytvořit například pomocí příkazu `makelist` nebo příkazy cyklu `for` - `do`.

Posloupnost (reálných čísel) je každá posloupnost $\{a_n\}_{n=1}^{\infty}$, jejíž členy jsou reálná čísla $a_n \in R$ (tj. zobrazení $N \to R$).

- **Explicitní zadání** (obecné vyjádření) člena a_n jako funkce proměnné n.
- **Rekurentní zadání** prvního člena a zadání a_n pomocí předchozích členů.

$\{a_n\}_{n=1}^{\infty} = \{2n-1\}_{n=1}^{\infty} = \{1, 3, 5, \ldots\}$.

- Explicitní zadání $a_n = 2n - 1$, $n \in N$.
- Rekurentní zadání $a_1 = 1$, $A_{n+1} = a_n + 2$, $n \in N$.

```
(%i3)  a(n):=2*n-1$ S:makelist(a(n),n,1,7);
(S)    [1, 3, 5, 7, 9, 11, 13]
(%i4)  an:1$ (for n:1 thru 7 do (print(an),an:an+2))$
       1
```

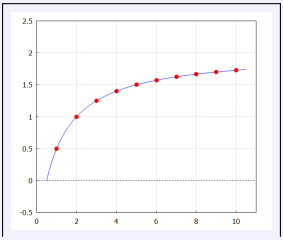
```
3
5
7
9
11
13
```

Příkaz `makelist` vytvoří seznam, který můžeme zobrazit i jako celek i po členech.

```
(%i2)  S1:makelist(2*n^2-1,n,1,10);S2:makelist(2*n^2-1,n,2,10,2);
(S1)   [1, 7, 17, 31, 49, 71, 97, 127, 161, 199]
(S2)   [7, 31, 71, 127, 199]
(%i4)  S1[1];S1[10];
(%o3)  1
(%o4)  199
```

Uspořádané dvojice se dávají do hranatých závorek a můžeme je zobrazovat jako body v rovině. V následujícím příkladu je vygenerována posloupnost i se svými vzory a následně vykreslena příkazem `draw2d`.

```
(%i1)  S1:makelist([n,(2*n-1)/(n+1)],n,1,10);
(S1)   [[1, 1/2], [2, 1], [3, 5/4], [4, 7/5], [5, 3/2], [6, 11/7], [7, 13/8], [8, 5/3], [9, 17/10], [10, 19/11]]
(%i2)  draw2d(grid=true,xaxis=true,yaxis=true,xrange=[0,11],yrange=[-0.5,2.5],
       color=blue,explicit((2*n-1)/(n+1),n,0.5,10.5),
       point_type=7,color=red,points(S1))$
```



Pomocí příkazů `for – do` vypíšeme několik členů posloupnosti $\left\{2n^2-1\right\}_{n=1}^{\infty}$.

```
(%i1)  (for n:1 thru 12 do (a_n: 2*n^2-1, print(a_n)) )$
       1
       7
       17
```

```
        31
        49
        71
        97
        127
        161
        199
        241
        287
```

Pěkným příkladem použití příkazů `for` – `do` je Fibonacciho posloupnost.

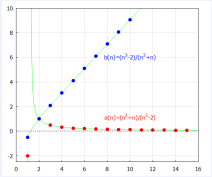
```
(%i3)   a0:0$ a1:1$ (for i:1 thru 12 do (an:a1+a0, print(an), a1:a0, a0:an))$
        1
        1
        2
        3
        5
        8
        13
        21
        34
        55
        89
        144
```

.....

- $\lim\limits_{n\to\infty} \frac{n^2+n}{n^3-2} = \lim\limits_{n\to\infty} \frac{n^3(n^{-1}+n^{-2})}{n^3(1-2n^{-3})} = \lim\limits_{n\to\infty} \frac{n^{-1}+n^{-2}}{1-2n^{-3}} = \frac{0+0}{1-0} = 0.$
- $\lim\limits_{n\to\infty} \frac{n^3-2}{n^2+n} = \lim\limits_{n\to\infty} \frac{n^2(n-2n^{-2})}{n^2(1+n^{-1})} = \lim\limits_{n\to\infty} \frac{n-2n^{-2}}{1+n^{-1}} = \frac{\infty-0}{1+0} = \infty.$

```
(%i1)   a(n):=(n^2+n)/(n^3-2)$ Sa:makelist([n,a(n)],n,1,15)$
        b(n):=(n^3-2)/(n^2+n)$ Sb:makelist([n,b(n)],n,1,15)$
        print("limit a(n)=",limit(a(n),n,inf),"  limit b(n)=",limit(b(n),n,inf))$
        draw2d(grid=true,xaxis=true,yaxis=true,xrange=[0,16],yrange=[-2.5,10],
        color=green,explicit(a(n),n,1,16),point_type=7,color=red,points(Sa),
          label(["a(n)=(n^2+n)/(n^3-2)",10,a(10)+1]),
        color=green,explicit(b(n),n,1,16),point_type=7,color=blue,points(Sb),
```

```
label(["b(n)=(n^3-2)/(n^2+n)",10,6]))$
```
(%o1) $limita(n) = 0 \quad limitb(n) = \infty$



Konečný i nekonečný součet vypočítáme pomocí příkazu `sum`.

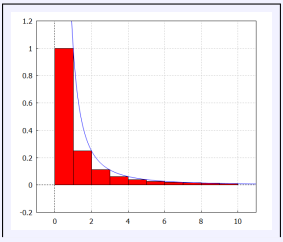
```
(%i1) sum(2*n^2-1,n,1,8);
```
(%o1) 400

Pomocí tohoto příkazu dokáže Maxima vypočítat přesný součet některých nekonečných řad. Součet řady můžeme zadat v menu `Calculus` a podmenu `Vypočítat Sum...`.

```
(%i2) sum(1/k^2,k,1,inf),simpsum; sum(1/k^2,k,1,inf);
```
(%o1) $\frac{\pi^2}{6}$

(%o2) $\sum_{k=1}^{\infty} \left(\frac{1}{k^2}\right)$

Číselná řada z předchozího příkladu může být graficky znázorněna následovně.

```
(%i1) a(n):=1/n^2$ rec:makelist(rectangle([i-1,0],[i,a(i)]),i,1,10)$
      draw2d(grid=true,xaxis=true,yaxis=true,xrange=[-1,11],yrange=[-0.2,1.2],
      border=true,color=black,fill_color=red,rec,
      color=blue,explicit(a(n),n,0,11))$
```


Číselné řady úzce souvisejí s posloupnostmi a zobecňují pojem sčítání na nekonečný počet sčítanců. Jednoduchým příkladem jsou zlomky a periodické čísla.

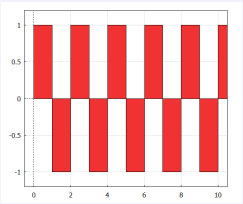
$\{a_n\}_{n=1}^{\infty}$ je posloupnost.

$$\Rightarrow \sum_{n=1}^{\infty} a_n = a_1 + a_2 + a_3 + \cdots + a_n + \cdots$$ se nazývá **(nekonečná číselná) řada**.

Pro nekonečné řady neplatí některá pravidla platná pro konečné počty sčítanců. Neplatí např. asociativní zákon:

$$\sum_{n=1}^{\infty} (-1)^{n+1} = \begin{cases} (1-1)+(1-1)+(1-1)+\cdots = 0+0+0+\cdots = 0, \\ 1+(-1+1)+(-1+1)+\cdots = 1+0+0+\cdots = 1. \end{cases}$$

```
(%i1) a(n):=(-1)^(n+1)$ rec:makelist(rectangle([i-1,0],[i,a(i)]),i,1,11)$
draw2d(grid=true,xaxis=true,yaxis=true,xrange=[-.5,10.5],yrange=[-1.2,1.2],
border=true,color=black,fill_color=red,rec)$
```



$\sum_{n=1}^{\infty} a_n$ je číselná řada.

- $s_k = \sum_{i=1}^{k} a_i = a_1 + a_2 + \cdots + a_k$, $k \in N$ se nazývá **k-tý částečný součet řady** $\sum_{n=1}^{\infty} a_n$.
- $r_k = \sum_{i=k+1}^{\infty} a_k = a_{k+1} + a_{k+2} + a_{k+3} + \cdots$ se nazývá **k-tý zbytek řady** $\sum_{n=1}^{\infty} a_n$.
- $\{s_k\}_{k=1}^{\infty} = \{s_n\}_{n=1}^{\infty}$ se nazývá **posloupnost částečných součtů řady** $\sum_{n=1}^{\infty} a_n$.

Vztah mezi $\sum_{n=1}^{\infty} a_n$ a posloupností $\{s_n\}_{n=1}^{\infty}$ je vzájemně jednoznačný.

Pro $\{s_n\}_{n=1}^{\infty}$ a $\sum_{n=1}^{\infty} a_n$ platí:

- $s_1 = a_1$.
- $a_1 = s_1 = s_1 - s_0$, kde $s_0 = 0$.
- $s_2 = a_1 + a_2 = s_1 + a_2$.
- $a_2 = s_2 - s_1$.
- $s_3 = a_1 + a_2 + a_3 = s_2 + a_3$.
- $a_3 = s_3 - s_2$.

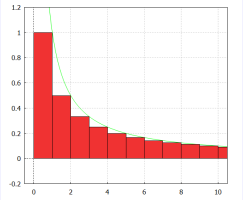
  ...
- $s_n = a_1 + a_2 + \cdots + a_{n-1} + a_n = s_{n-1} + a_n$.
- $a_n = s_n - s_{n-1}$, $n \in N$.

Součet řady $\sum_{n=1}^{\infty} a_n$ se nazývá $\lim_{n\to\infty} s_n = s \in R^*$ (pokud existuje), označení $\sum_{n=1}^{\infty} a_n = s$.

Harmonická řada

$\sum_{n=1}^{\infty} \frac{1}{n} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \cdots = \infty.$

```
(%i1) a(n):=1/n$ rec:makelist(rectangle([i-1,0],[i,a(i)]),i,1,11)$
draw2d(grid=true,xaxis=true,yaxis=true,xrange=[-.5,10.5],yrange=[-.2,1.2],
color=green,explicit(a(n),n,.5,11),
border=true,color=black,fill_color=light_red,rec)$
```

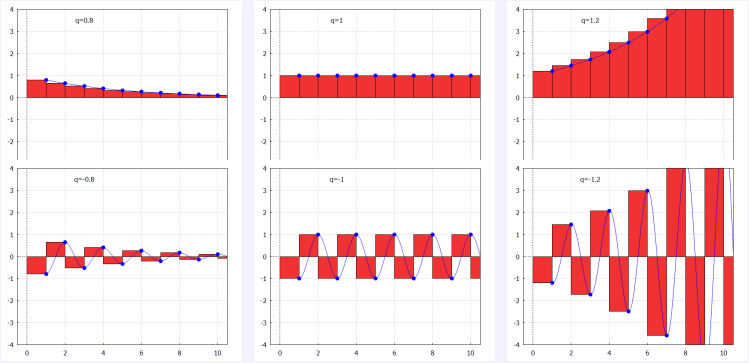


Geometrická řada

$\sum_{n=1}^{\infty} q^{n-1} = 1 + q + q^2 + \cdots = \frac{1}{1-q}$ pro všechny $q \in (-1; 1)$.

V následujícím příkladu stačí měnit na začátku hodnotu q.

```
(%i1) q:0.8$ a(n,q):=q^n$ peca:makelist([i,a(i,q)],i,1,11)$
reca:makelist(rectangle([i-1,0],[i,a(i,q)]),i,1,11)$
draw2d(grid=true,xaxis=true,yaxis=true,xrange=[-.5,10.5],yrange=[-4,4],
border=true,color=black,fill_color=light_red,reca,
label([concat("q=",string(q)),3,3.5]), color=blue,explicit(a(n,q),n,1,11),
point_type=7,color=blue,points(peca))$
```




```
(%i4) sq(q):=sum(q^n,n,1,inf)$
      sq(1/2),simpsum; sq(1/3),simpsum; sq(-1/2),simpsum; sq(2),simpsum;
(%i1) 1
(%i2) 1/2
(%i3) -1/3
(%i4) sum: sum is divergent.
```

3 Funkce

Funkce $y = f(x)$, $x \in D(f)$, tj. $f\colon D(f) \to H(f)$.

- Množina $\{[x;y] \in R^2; x \in D(f),\ y = f(x)\}$ se nazývá **graf funkce** f.
- **Funkce reálné proměnné**, jestliže definiční obor $D(f) \subset R$.
- **Reálná funkce**, pokud obor hodnot $H(f) \subset R$.

$y = f(x)$, $x \in A$ se nazývá:

- **Injektivní (injekce, prostá)**, jestliže pro všechny $x_1, x_2 \in A$, $x_1 \neq x_2$ platí $f(x_1) \neq f(x_2)$,

  tj. z rovnosti $f(x_1) = f(x_2)$ plyne rovnost $x_1 = x_2$.

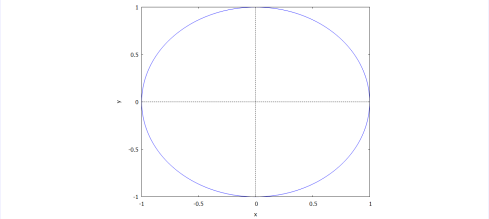
- **Surjektivní (surjekce, na množinu)**, pokud $f(A) = B$,
  tj. ke každému $y \in B$ existuje $x \in A$ takové, že $y = f(x)$.
- **Bijektivní (bijekce)**, pokud je injektivní a surjektivní.

$y = f(x)$, $x \in D(f)$ se vyjadřuje:

- **Explicitně**, tj. analyticky vzorcem $y = f(x)$, $x \in D(f)$.
- **Parametricky** rovnicemi $x = \varphi(t)$, $y = \psi(t)$, $t \in J$, $J \subset R$, kde $\varphi, \psi\colon J \to R$.
  Parametr t má pomocný význam.
- **Implicitně** rovnicí $f(x, y) = 0$, kde $f\colon R^2 \to R$ a podmínkami pro $[x; y]$.

Pokud chceme v programu Maxima zobrazit funkci zadanou implicitně, musíme načíst knihovnu `implicit_plot`.

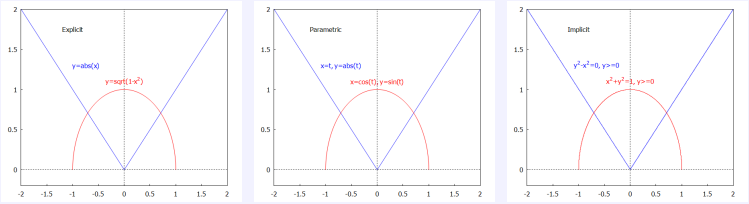
```
(%i1)  load(implicit_plot);
(%o1)  ../share/contrib/implicit_plot.lisp
(%i2)  implicit_plot(x^2+y^2-1,[x,-1,1],[y,-1,1])$
       implicit_plot is now obsolete. Using plot2d instead:
           plot2d (y^2+x^2-1=0,[x,-1,1],[y,-1,1])
(%i2)  plot2d(x^2+y^2-1=0,[x,-1,1],[y,-1,1])$     /* is correct */
```



Funkci f: $y = |x|$, $x \in R$ můžeme zadat např.:

- Explicitně: $y = \sqrt{x^2}$ resp. $y = \max\{-x, x\}$.
- Parametricky: $x = t$, $y = |t|$ $t \in R$, resp. $x = t$, $y = \sqrt{t^2}$, $t \in R$.
- Implicitně: $y^2 - x^2 = 0$, $y \geq 0$, resp. $y - |x| = 0$.

```
(%i1)  load(implicit_plot)$
(%i2)  draw2d(xaxis=true, yaxis=true, xrange=[-2,2], yrange=[-.2,2],
       color=blue, explicit(abs(x),x,-2,2), label(["y=abs(x)",-.75,1.3]),
       color=red, explicit(sqrt(1-x^2),x,-1,1), label(["y=sqrt(1-x^2)",0,1.1]),
       color=black, label(["Explicit",-1,1.75]))$
(%i3)  draw2d(xaxis=true, yaxis=true, xrange=[-2,2], yrange=[-.2,2],
       color=blue, parametric(t,abs(t),t,-2,2), label(["x=t, y=abs(t)",-.7,1.3]),
       color=red, nticks=100, parametric(cos(t),sin(t),t,0,%pi),
       label(["x=cos(t), y=sin(t)",0,1.1]),
       color=black, label(["Parametric",-1,1.75]))$
(%i4)  draw2d(xaxis=true, yaxis=true, xrange=[-2,2], yrange=[-.2,2],
       color=blue, implicit(y^2-x^2,x,-2,2,y,0,2), label(["y^2-x^2=0, y>=0",-.65,1.3]),
       color=red, implicit(x^2+y^2-1,x,-1,1,y,0,1), label(["x^2+y^2=1, y>=0",0,1.1]),
       color=black, label(["Implicit",-1,1.75]))$
```



4 Průběh funkce

Důležitou součástí vyšetřování průběhu funkce je určení intervalů, na kterých je tato funkce monotónní.

Vyšetřit průběh funkce f znamená určit:

- Definiční obor $D(f)$, body a intervaly spojitosti a nespojitosti.
- Sudost, lichost, periodicitu, resp. jiné speciální vlastnosti.
- Jednostranné limity v bodech nespojitosti, v hraničních bodech a v bodech $\pm\infty$.
- Nulové body; intervaly, na kterých je f kladná a záporná.
- f', stacionární body, lokální a globální extrémy; intervaly, na kterých je f rostoucí, klesající a konstantní.
- f'', inflexní body; intervaly, na kterých je f konvexní a konkávní.
- Asymptoty bez směrnice a asymptoty se směrnicí.
- Obor hodnot $H(f)$ a nastínit graf funkce.

Nejnázornější představu o průběhu funkce nám většinou poskytne graf. Při jeho konstrukci využíváme všechny zjištěné údaje. Mnohokrát jsou ale nedostatečné, proto je musíme doplnit vhodně zvolenými funkčními hodnotami.

Průběh funkce $f(x) = \frac{8(x-2)}{x^2} = \frac{8x-16}{x^2}$.

```
(%i1)  f(x):=(8*x-16)/x^2;
(%o1)  f(x) := (8x-16)/x^2
```

- $D(f) = R - \{0\} = (-\infty; 0) \cup (0; \infty)$.

Pomocí příkazu `denom` (denominator) zjistíme, kdy je jmenovatel nulový.

```
(%i3)  fm:denom(f(x));solve(fm=0,x);
(fmen) x^2
(%o3)  [x = 0]
```

- f není periodická, f není sudá, f není lichá.
- f je spojitá na intervalech $(-\infty; 0)$, $(0; \infty)$, v bodě 0 je nespojitá.
- $\lim\limits_{x\to\pm\infty} f(x) = \lim\limits_{x\to\pm\infty} \frac{8x-16}{x^2} = \lim\limits_{x\to\pm\infty} \left(\frac{8}{x} - \frac{16}{x^2}\right) = \frac{8}{\pm\infty} - \frac{16}{\infty} = 0 - 0 = 0.$

```
(%i5)  limit(f(x),x,minf);limit(f(x),x,inf);
(%o4)  0
(%o5)  0
```

- $\lim\limits_{x\to 0^-} f(x) = \lim\limits_{x\to 0^-} \frac{8(x-2)}{x^2} = \frac{-16}{0^+} = -\infty$, $\lim\limits_{x\to 0^+} f(x) = \lim\limits_{x\to 0^+} \frac{8(x-2)}{x^2} = \frac{-16}{0^+} = -\infty$.

```
(%i7)  limit(f(x),x,0,minus);limit(f(x),x,0,plus);
(%o6)  -inf
(%o7)  -inf
```

- Bod $x = 0$ je neodstranitelný bod nespojitosti II. Druhu.
- $x = 0$ je asymptota bez směrnice.
- $f(x) = \frac{8x-16}{x^2} = 0. \Leftrightarrow 8x - 16 = 0. \Leftrightarrow x = 2.$

Pomocí příkazu `num` (numerator) zjistíme, kdy je čitatel nulový.

```
(%i9)  fcit:num(f(x));solve(fcit=0,x);
(fcit) 8x − 16
(%o9)  [x = 2]
```

- $x = 2$ je nulový bod f. $\Rightarrow$ $\begin{cases} f(x) < 0 \text{ pro } x \in (-\infty; 0), \\ f(x) < 0 \text{ pro } x \in (0; 2), \\ f(x) > 0 \text{ pro } x \in (2; \infty). \end{cases}$

$f(2) = 0$, f není v bodě $x = 0$ definována.

$\Rightarrow$ Funkce f nemění znaménko na intervalech $(-\infty; 0)$, $(0; 2)$, $(2; \infty)$.

$\Rightarrow$ Stačí zvolit libovolný bod v daných intervalech a ověřit jeho hodnotu.

```
(%i13) f(2);f(-1);f(1);f(3);
(%o10) 0
(%o11) −24
(%o12) −8
(%o13) 8/9
```

- $f'(x) = \left[\frac{8x-16}{x^2}\right]' = \frac{8x^2-(8x-16)2x}{x^4} = \frac{32x-8x^2}{x^4} = \frac{32-8x}{x^3}$, $x \in R$, $x \neq 0$.

```
(%i15) f1(x):=diff(f(x),x,1)$ ratsimp(f1(x));
(%o15) −(8x−32)/x^3
```

- $f'(x) = \frac{32-8x}{x^3} = 0. \Leftrightarrow 32 - 8x = 0. \Leftrightarrow x = 4.$

```
(%i16) solve(f1(x)=0,x);
(%o16) [x = 4]
```

- f' je v bodě 0 nespojitá.

```
(%i18) f1men:denom(ratsimp(f1(x)));solve(f1men=0,x);
(f1men) x^3
(%o18) [x = 0]
```

- $x = 4$ je nulový bod f'. $\Rightarrow \begin{cases} f'(x) < 0,\ f \text{ je klesající pro } x \in (-\infty; 0), \\ f'(x) > 0,\ f \text{ je rostoucí pro } x \in (0; 4), \\ f'(x) < 0,\ f \text{ je klesající pro } x \in (4; \infty). \end{cases}$

$f'(4) = 0$, f' není v bodě $x = 0$ definována.

$\Rightarrow$ Funkce f' nemění znaménko na intervalech $(-\infty; 0)$, $(0; 4)$, $(4; \infty)$.

$\Rightarrow$ Stačí zvolit libovolný bod v daných intervalech a ověřit jeho hodnotu.

```
(%i22) subst(4,x,f1(x));subst(-1,x,f1(x));subst(1,x,f1(x));subst(5,x,f1(x));
(%o19) 0
(%o20) -40
(%o21) 24
(%o22) -8/125
```

- f má v bodě $x = 4$ lokální maximum i globální maximum $f(4) = 1$.

```
(%i23) f(4);
(%o23) 1
```

- f nemá lokální a ani globální minimum.
- $f''(x) = \left[\frac{32-8x}{x^3}\right]' = \frac{-8x^3-(32-8x)3x^2}{x^6} = \frac{16x^3-96x^2}{x^6} = \frac{16x-96}{x^4}$, $x \in R$, $x \neq 0$.

```
(%i25) f2(x):=diff(f(x),x,2)$ ratsimp(f2(x));
(%o25) (16*x-96)/x^4
```

- $f''(x) = \frac{16x-96}{x^4} = 0. \Leftrightarrow 16x - 96 = 0. \Leftrightarrow x = 6.$

```
(%i26)  solve(f2(x)=0,x);
(%o26)  [x = 6]
```

- f'' je v bodě 0 nespojitá.

```
(%i28)  f2men:denom(ratsimp(f2(x)));solve(f2men=0,x);
(f2men)  x^4
(%o28)  [x = 0]
```

- $x = 6$ je nulový bod f''. $\Rightarrow$ $\begin{cases} f''(x) < 0,\ f \text{ je konkávní pro } x \in (-\infty; 0), \\ f''(x) < 0,\ f \text{ je konkávní pro } x \in (0; 6), \\ f''(x) > 0,\ f \text{ je konvexní pro } x \in (6; \infty). \end{cases}$

$f'(6) = 0$, f'' není v bodě $x = 0$ definována.

$\Rightarrow$ Funkce f'' nemění znaménko na intervalech $(-\infty; 0)$, $(0; 6)$, $(6; \infty)$.

$\Rightarrow$ Stačí zvolit libovolný bod v daných intervalech a ověřit jeho hodnotu.

```
(%i32)  subst(6,x,f2(x));subst(-1,x,f2(x));subst(1,x,f2(x));subst(7,x,f2(x));
(%o29)  0
(%o30)  −112
(%o31)  −80
(%o32)  16/2401
```

- Bod $x = 6$ je inflexní bod funkce f.

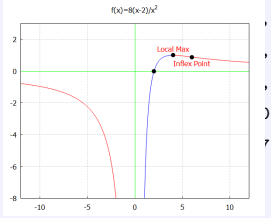
```
(%i33)  f(6);
(%o33)  8/9
```

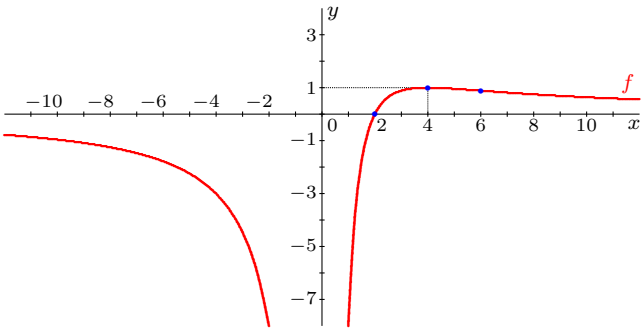
$\left.\begin{array}{l} \bullet\ k = \lim\limits_{x \to \pm\infty} \frac{f(x)}{x} = \lim\limits_{x \to \pm\infty} \frac{8x-16}{x^3} = \lim\limits_{x \to \pm\infty} \left(\frac{8}{x^2} - \frac{16}{x^3}\right) = 0 - 0 = 0. \\ \bullet\ q = \lim\limits_{x \to \pm\infty} [f(x) - kx] = \lim\limits_{x \to \pm\infty} [f(x) - 0 \cdot x] = \lim\limits_{x \to \pm\infty} f(x) = 0. \end{array}\right\} \Rightarrow y = kx + q = 0.$

```
(%i35) km:limit(f(x)/x,x,minf);kp:limit(f(x)/x,x,inf);
(km)   0
(kp)   0
(%i37) qm:limit(f(x)-km*x,x,minf);qp:limit(f(x)-kp*x,x,inf);
(km)   0
(kp)   0
```

- $y = 0$ je asymptota se směrnicí.
- $H(f) = (-\infty; 1\rangle$.

```
(%i38) draw2d(grid=true,xaxis=true,yaxis=true,xrange=[-12,12],yrange=[-8,3],
                                                  ,color=blue,explicit(f(x),x,0,4),
                                                  ,x,-12,0),explicit(f(x),x,4,12),
                                                  ,f(6)-.4],["Local Max",4,f(4)+.4]),
                                                  ),t,t,-8,3),parametric(t,0,t,-12,12),
                                                  ',points([[4,f(4)],[6,f(6)],[2,f(2)]]))$
```





Obr. 1: *Graf funkce* $f(x) = \frac{8(x-2)}{x^2}$

Literatura

1. Blaško R., *Matematická analýza I,* Žilina, EDIS 2009.
2. Blaško R., *Matematická analýza I,* skriptum, `http://frcatel.fri.utc.sk/~beerb/ma1/sa1.pdf`.

3. Blaško R., *Neurčitý a určitý integrál reálnej funkcie,* skriptum, http://frcatel.fri.utc.sk/~beerb/ma1/sa2.pdf.

4. Blaško R., *Základy lineárnej algebry a základy matematickej analýzy pre manažérov,* skriptum, http://frcatel.fri.utc.sk/~beerb/ma1/zla-zma.pdf.

5. Buša J., *Maxima Open source systém počítačovej algebry,* online, http://people.tuke.sk/jan.busa/kega/maxima/maxima.pdf, 2006.

6. Bittinger M. L., Ellenbogen D. J., Surgent S. A., *Calculus and its Applications,* Addisok-Wesley, ISBN-10: 0-321-69433-3.

7. Crowell B., *Calculus, Light and Matter,* www.lightandmatter.com, March 2010.

8. Hannan Z., *wxMaxima for Calculus I and II,* Solano Community College, https://wxmaximafor.wordpress.com/.

9. Mardsen J., Weinstein A., *Calculus I–III,* Springer.

10. Strang G., Calculus, *Wellesley-Cambridge Press,* Box 82-279 Wellesley MA 02181.

Autor

Rudolf Blaško, Katedra matematických metód a operačnej analýzy, Fakulta riadenia a informatiky, Žilinská univerzita v Žiline, Vysokoškolákov 8215/1, 010 26 Žilina, Slovenská republika, e-mail: beerb@frcatel.fri.uniza.sk