Laborator 1: Introducere in SageMath

Calcule in SageMath

Operatii algebrice de baza

```
“cele patru operatii”     a+b, a-b, a*b, a/b

ridicarea la putere                 a^b sau a**b

radical           sqrt(a)

radical de ordin n             a^(1/n)
```

In notebook, se pot introduce comenzi in linia de comanda, rezultatul obtinandu-se apasand butonul Run sau folosind combinatia de taste Shift+Enter

Combinatia de taste Alt+Enter executa comanda si insereaza o noua linie de comanda.

In [1]:
```
1+2-3
```

Out[1]:
```
0
```

In [2]:
```
2*3/7+3^2
```

Out[2]:
```
69/7
```

In [3]:
```
(1 + 2*(3 + 5^2))*sqrt(9)
```

Out[3]:
```
171
```

Pentru a obtine valoarea aproximativa a unui numar, se va scrie acel numar urmat de simbolul "." (punctul decimal)

In [4]:
```
15/6
```

Out[4]:
```
5/2
```

In [5]:
```
15./6
```

Out[5]:
```
2.50000000000000
```

De asemenea, functia *numerical_approx* returneaza valoarea aproximativa a unei expresii numerice

In [6]:
```
numerical_approx(15/6)
```

Out[6]:
```
2.50000000000000
```

In [7]:
```
 numerical_approx(44/13, digits=60)
```

Out[7]:
```
3.38461538461538461538461538461538461538461538461538461538462
```

Alte operatii cu numere intregi

```
catul impartirii                   a // b

restul impartirii                  a % b

catul si restul                    divmod(a,b)

n!                                 factorial(n)

coeficient binomial                binomial(n,k)
```

Functii matematice uzuale

```
parte intreaga                    floor(a)

modulul                           abs(a)

exponentiala si logaritmul        exp, log

Logaritm in baza a                log(x, a)

Functii trigonometrice            sin, cos, tan

Functii trigonometrice inverse    arcsin, arccos, arctan

Functii hiperbolice               sinh, cosh, tanh

Functii hiperbolic inverse        arcsinh, arccosh, arctanh

Partea intreaga, etc              floor, ceil, trunc, round

Radical de ordin 2 si n           sqrt, nth_root
```

Variabile

Variabilele trebuie declarate explicit inainte de a fi utilizate. (SR este abrevierea de la Symbolic Ring):

In [8]:
```
x = SR.var('x')
p1=(x+1)^2
p1
```

Out[8]:
```
(x + 1)^2
```

Pentru a dezvolta expresia se foloseste comanda *variable.expand()*

In [9]:
```
p1.expand()
```

Out[9]:
```
x^2 + 2*x + 1
```

In [10]:
```
p2=x^2 + 2*x + 1
p2
```

Out[10]:
```
x^2 + 2*x + 1
```

Pentru a descompune in factori o expresie data se foloseste comanda *variabila.factor()*

In [11]:
```
p2.factor()
```

Out[11]:
```
(x + 1)^2
```

In [12]:
```
p3=x^2-5*x + 6
p3.factor()
```

Out[12]:
```
(x - 2)*(x - 3)
```

Comanda *vaiabila.subs()* se foloseste pentru a calcula valoarea unei expresii pentru o anumita valoare a parametrilor sai.

In [13]:
```
p1.subs(x=-1)
```

Out[13]:
```
0
```

In [14]:
```
p3.subs(x=2)
```

Out[14]:
```
0
```

In [15]:
```
p3.subs(x=-3)
```

Out[15]:
```
30
```

Comanda *var('x')* se poate folosi in locul comenzii *x = SR.var('x')*

In [16]:
```
x=var('x')
p=(2*x-1)^3
p.expand()
```

Out[16]:
```
8*x^3 - 12*x^2 + 6*x - 1
```

In [17]:
```
x,y=var('x,y')
p=(2*x+y)^2
p.expand()
```

Out[17]:
```
4*x^2 + 4*x*y + y^2
```

In [18]:
```
p.subs(x=1,y=1)
```

Out[18]:
```
9
```

Ecuatii si sisteme de ecuatii algebrice

O ecuatie se defineste folosind semnul "==", ex. *f(x)==g(x)*.

Cele mai folosite comenzi pentru rezolvarea ecuatiilor sunt:

```
Solutie simbolica                          solve

Radacini (cu ordin de multiplicitate)      roots

Solutie numerica                           find_root
```

In [19]:
```
x=var('x')
eq1=x^2+x+1==0
solve(eq1,x)
```

Out[19]:
```
[x == -1/2*I*sqrt(3) - 1/2, x == 1/2*I*sqrt(3) - 1/2]
```

Nu toate ecuatiile pot fi rezolvate cu ajutorul softului SageMath, in exemplul urmator SageMath nu returneaza nicio solutie.

In [20]:
```
eq2=exp(-x)==x
solve(eq2,x)
```

Out[20]:
```
[x == e^(-x)]
```

Pentru a determina o solutie numerica se foloseste *find_root(equation,a,b)*, comanda ce va determina solutia in intervalul *[a,b]*

In [21]:
```
find_root(eq2,0,2)
```

Out[21]:
```
0.5671432904098384
```

Comanda *solve* poate fi folosita si pentru sisteme de ecuatii, acestea putand fi definite in Sage folosind [ ]

[eq1,eq2,...,eqn].

In [22]:
```
x,y=var('x,y')
syst=[x+2*y==1,x-y==3]
solve(syst,x,y)
```

Out[22]:
```
[[x == (7/3), y == (-2/3)]]
```

Limite

Pentru a calcula o limita se va folosi comanda *limit*

In [23]:
```
n,x=var('n,x')
```

In [24]:
```
limit(1/n,n=infinity)
```

Out[24]:
```
0
```

In [25]:
```
limit(sin(x)/x,x=0)
```

Out[25]:
```
1
```

In [26]:
```
limit(1/x, x=0)
```

Out[26]:
```
Infinity
```

Ultimul rezultat se refera la faptul ca una dintre limite, la stanga sau la dreapta, este infinita. Pentru a calcula limita la stanga (minus) sau la dreapta (plus) se poate folosi optiunea *dir*.

In [27]:
```
limit(1/x, x=0,dir='minus')
```

Out[27]:
```
-Infinity
```

In [28]:
```
limit(1/x, x=0,dir='plus')
```

Out[28]:
```
+Infinity
```

Functii folositoare la analiza:

```
Derivata                    diff(f(x), x)

Derivata de ordin n         diff(f(x), x, n)

Primitiva                   integrate(f(x), x)
```

Integrala definita	integral_numerical(f(x), a, b)
Suma	sum(f(i), i, imin, imax)
Limita	limit(f(x), x=a)
Serie Taylor	taylor(f(x), x, a, n)
Serie de puteri	f.series(x==a, n)

Derivarea functiilor

In [29]:
```
f(x)=exp(x^2)+3
```

In [30]:
```
diff(f(x),x,2)
```

Out[30]:
```
4*x^2*e^(x^2) + 2*e^(x^2)
```

In [31]:
```
diff(f(x),x,3)
```

Out[31]:
```
8*x^3*e^(x^2) + 12*x*e^(x^2)
```

Integrarea functiilor

In [32]:
```
x=var('x')
```

In [33]:
```
integrate(cos(x),x)
```

Out[33]:
```
sin(x)
```

In [34]:
```
f(x)=exp(x)*cos(x)
integrate(f(x),x)
```

Out[34]:
```
1/2*(cos(x) + sin(x))*e^x
```

Pentru integrale definita se foloseste *integrate(f(x),x,a,b)*

In [35]:
```
integrate(cos(x),x,0,pi/2)
```

Out[35]:
```
1
```

Sage nu poate calcula intotdeauna valoarea integralei definite

In [36]:
```
integrate(sin(sqrt(1 - x^3)), x, 0,1)
```

Out[36]:
```
integrate(sin(sqrt(-x^3 + 1)), x, 0, 1)
```

Pentru a determina valoarea numerica a unei intergrale definite pe un interval se foloseste functia *integral_numerical*, care returneaza o pereche, in care prima valoare este valoarea aproximativa integralei, iar a doua pereche este eroarea estimarii.

In [37]:
```
integral_numerical(sin(sqrt(1 - x^3)), 0, 1)
```

Out[37]:
```
(0.7315380084233594, 3.953379981670976e-07)
```

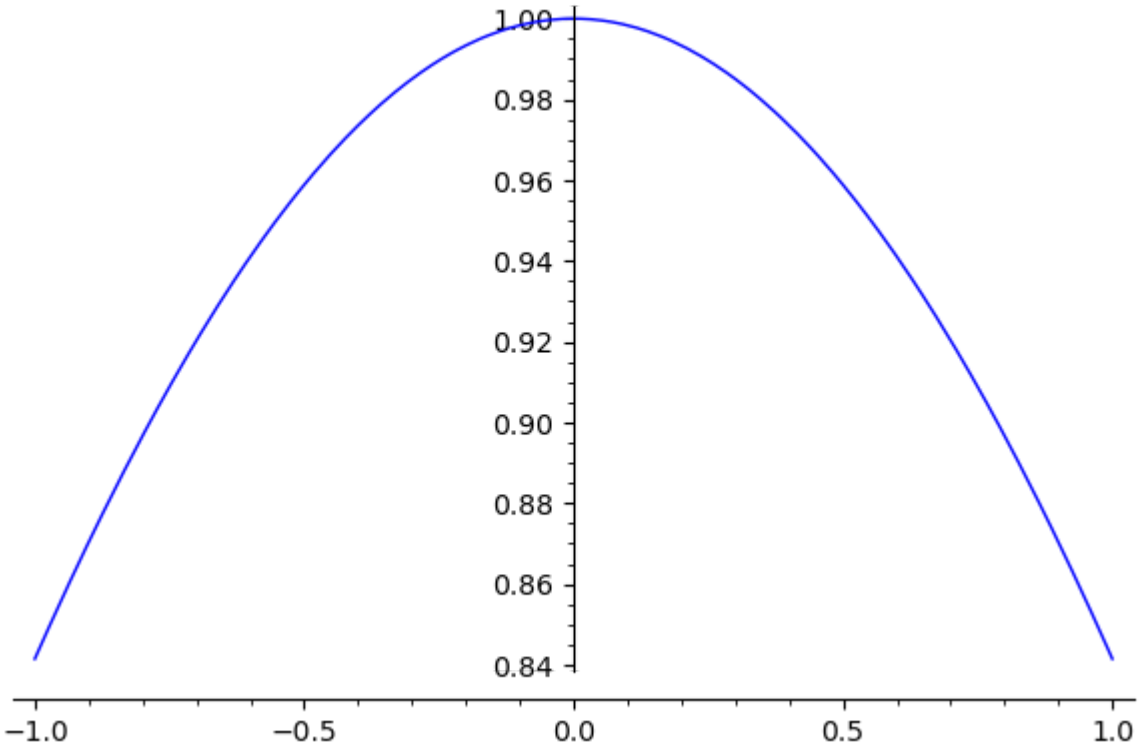
Grafice 2D

Reprezentarea grafica a functiilor

Pentru a reprezenta grafic functia $f(x)$ pe intervalul $[a, b]$, se foloseste comanda *plot(f(x), a, b)* sau sintaxa alternativa *plot(f(x), x, a, b)*.

In [38]:
```
f(x)=sin(x)/x
plot(f(x),-1,1)
```

Out[38]:



Comanda plot are mai multe optiuni. Cele mai importante sunt:

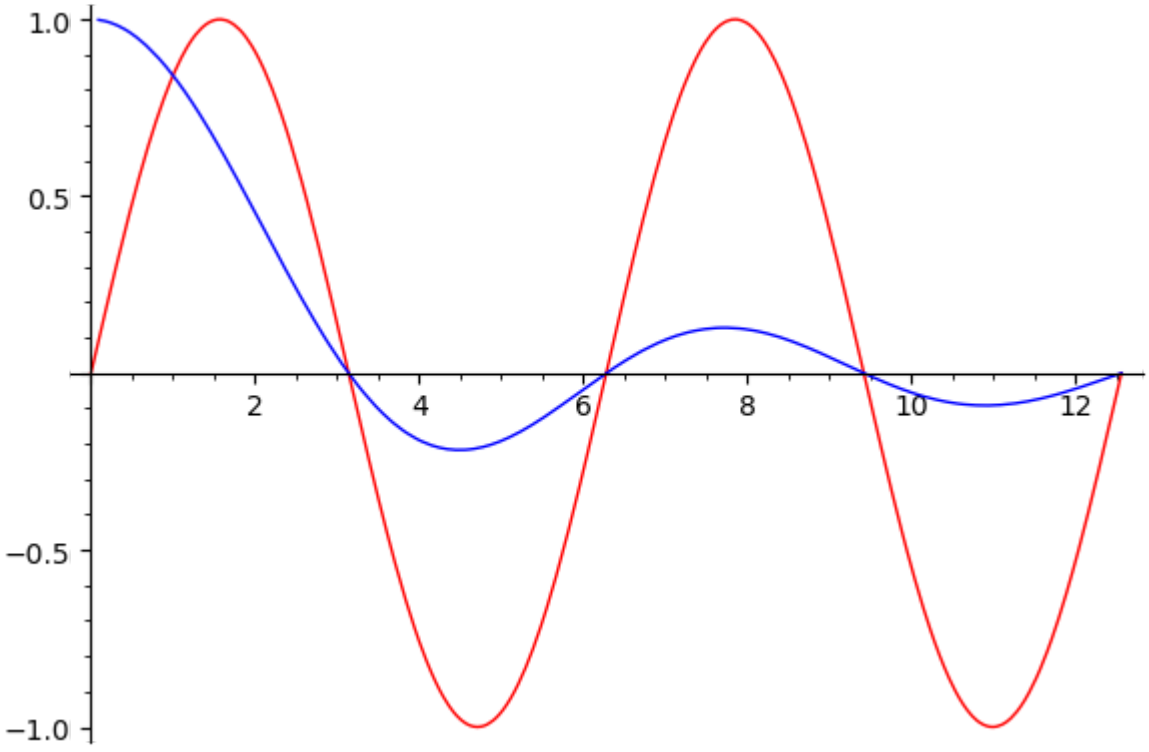
```
plot_points (default value 200): numarul minim de puncte calculate;
```

```
xmin and xmax: capetele intervalului de reprezentare a functiei;

color: culoarea graficului, sau tripletul RGB corespunzator culorii, sau
numele culorii cum ar fi 'blue', sau codul HTML al culorii cum ar fi
'#aaff0b';

detect_poles (default value False): determina punctele de discontinuitate;

alpha: transparenta culorii liniei;

thickness: grosimea liniei;

linestyle: stilul graficului utilizat cum ar fi grafic prin linie punctata
':', grafic prin linii intrerupte  '-.', sau grafic prin linie continua
(valoarea implicita) '-'.
```

Se pot reprezenta grafic mai multe functii in aceeasi fereastra, specificand lista functiilor *[f1(x),f2(x),...,fn(x)]* si lista corespunzatoare a culorilor *['color_1','color_2','color_n']*:

In [39]:
```
plot([sin(x),f(x)],0,4*pi,color=['red','blue'])
```

Out[39]:



Daca lista de functii este mare, se poate folosi comanda *for* pentru a genera respectiva lista prin indexarea elementelor listei. De exemplu, pentru functia $f_n(x) = \frac{x}{(1+x^2)^n}$ daca dorim sa reprezentam grafic a functiile $f_1(x)$, ..., $f_{10}(x)$ vom construi prima data lista indexand functiile dupa n si apoi vom reprezenta garficele folosind comanda *plot*:

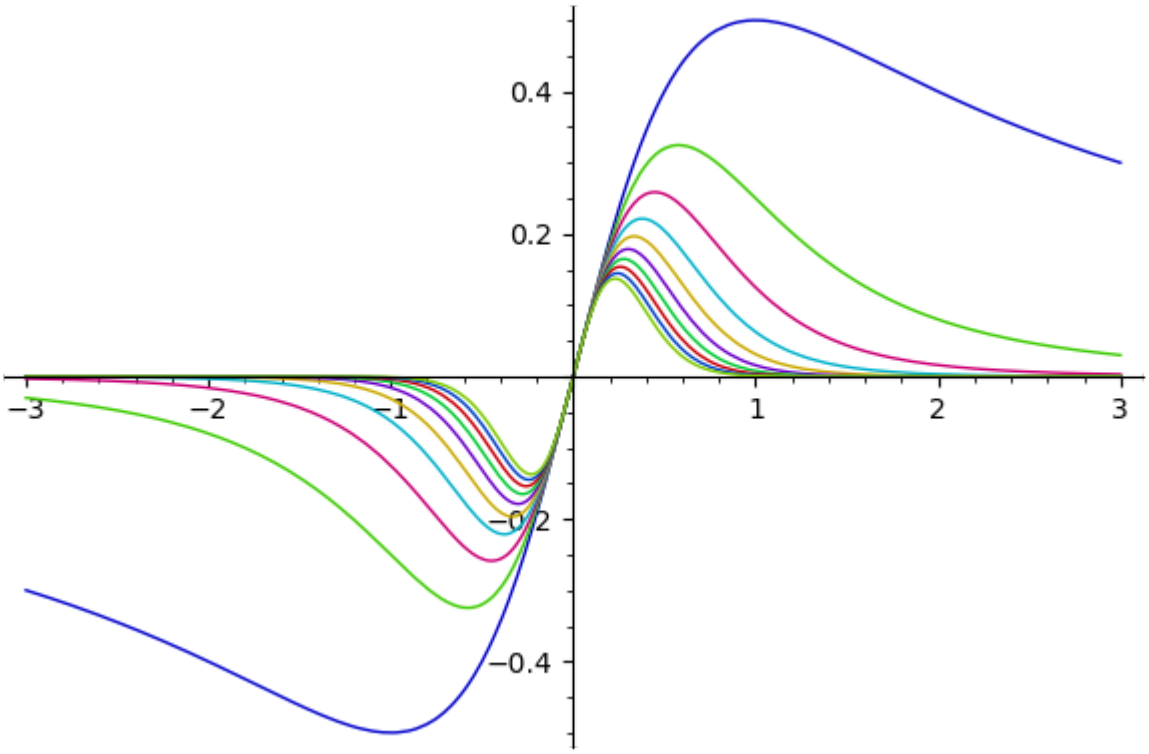
In [40]:
```
x,n=var('x,n')
f(x,n)=x/(1+x^2)^n
```

In [41]:
```
f_list=[f(x,n) for n in [1..10]]
f_list
```

Out[41]:
```
[x/(x^2 + 1),
 x/(x^2 + 1)^2,
 x/(x^2 + 1)^3,
 x/(x^2 + 1)^4,
 x/(x^2 + 1)^5,
 x/(x^2 + 1)^6,
 x/(x^2 + 1)^7,
 x/(x^2 + 1)^8,
 x/(x^2 + 1)^9,
 x/(x^2 + 1)^10]
```

In [42]:
```
plot(f_list,-3,3)
```

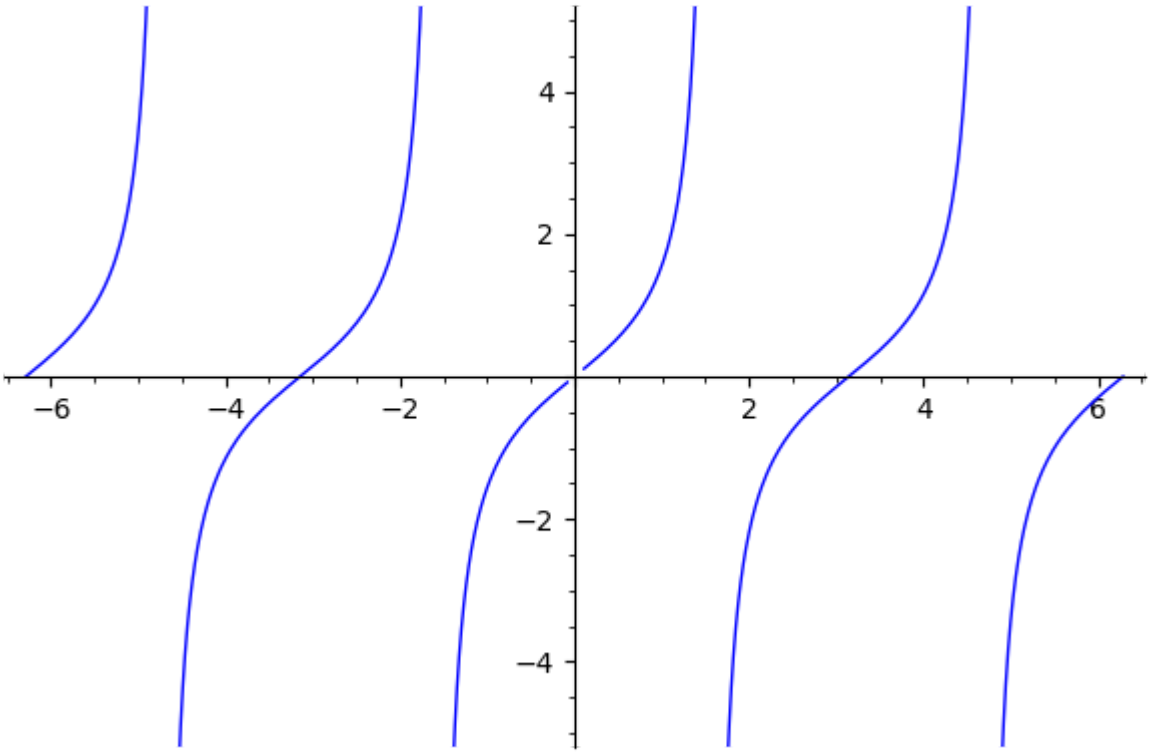
Out[42]:



In cazul in care exista puncte de discontinuitate se poate folosi optiunea *detect_poles=True* si putem alege intervalul de marginire a functiei, precizind *ymin* si *ymax* (intervalul pe axa 0y)

In [43]:
```
plot(tan(x), -2*pi,2*pi,detect_poles=True,ymin=-5,ymax=5)
```

Out[43]:



Curbe in forma parametrica

Daca o curba in forma parametrica este data de

$$\begin{cases} x(t) &= f(t) \\ y(t) &= g(t) \end{cases}, \quad t \in [a, b]$$

se va folosi comanda *parametric_plot((f(t), g(t)), (t, a, b))*.

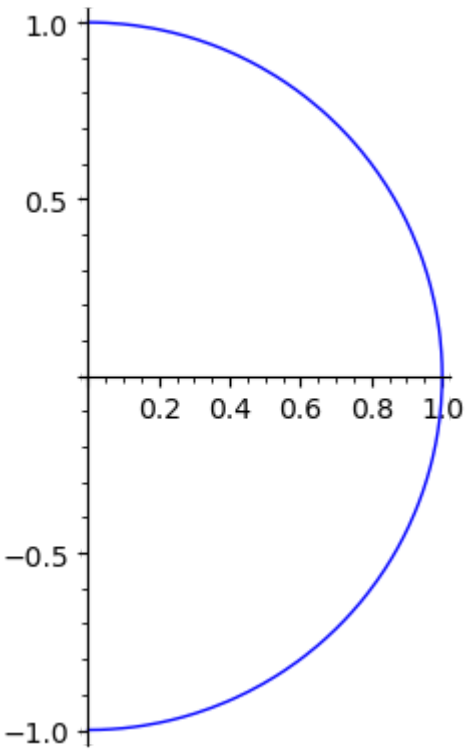
De exemplu, in cazul semicercului definit de

$$\begin{cases} x(t) &= \cos(t) \\ y(t) &= \sin(t) \end{cases}, \quad t \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$$

avem:

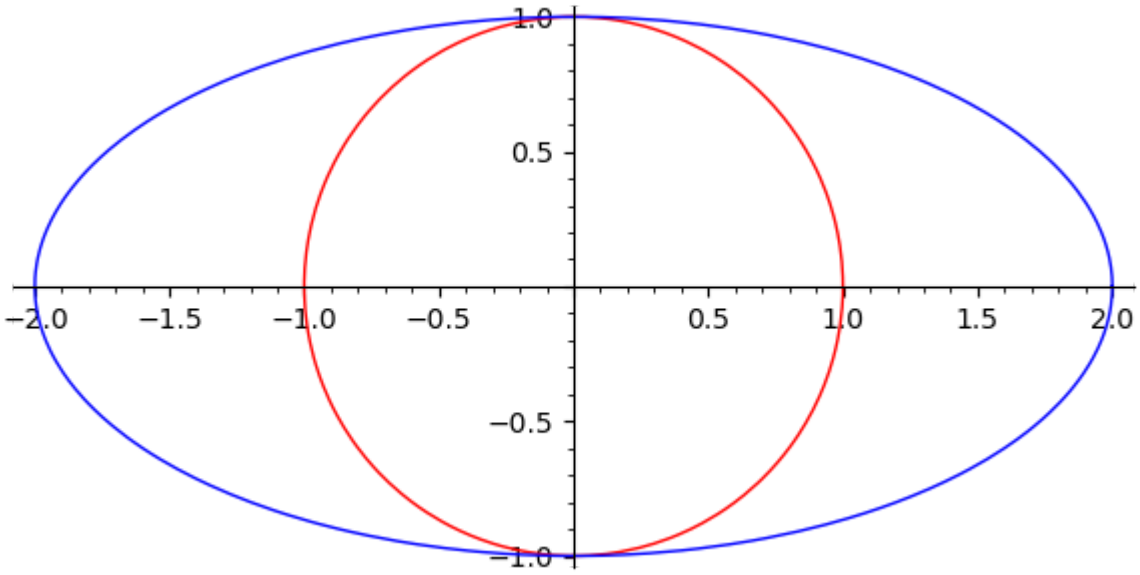
In [44]:
```
t=var('t')
x(t)=cos(t)
y(t)=sin(t)
parametric_plot((x(t), y(t)), (t, -pi/2, pi/2))
```

Out[44]:



Pentru a reprezenta in acelasi sistem de coordonate mai multe curbe date in forma parametrica, vom atribui fiecarui grafic o variabila si le vom combina folosing comanda plus (+), iar pentru afisare comanda *show*.

In [45]:
```
g1=parametric_plot((x(t), y(t)), (t, -pi/2, 3*pi/2),color='red')
g2=parametric_plot((2*x(t), y(t)), (t, 0, 2*pi),color='blue')
show(g1+g2)
```

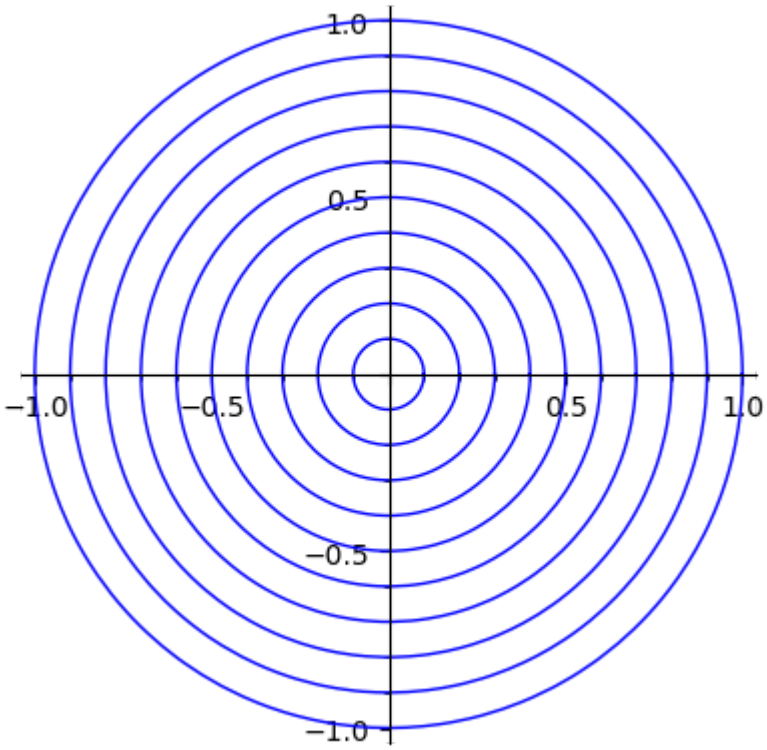


Când lista de curbe date prin ecuatii parametrice este mare, putem folosi comanda *for* pentru a genera lista de curbe si prin adunare le combinam in acelasi grafic, de exemplu, sa reprezentam familia cercurilor centrate in origine $(0,0)$ si de raza r

$$\begin{cases} x(t) &= r\cdot\cos(t) \\ y(t) &= r\cdot\sin(t) \end{cases}, \quad t\in[0,2\pi]$$

pentru $r=0.1,0.2,\ldots,1$

In [46]:
```
g=parametric_plot((1/10*cos(t),1/10*sin(t)),(t,0,2*pi))
for k in [2..10]:
    g1=parametric_plot((k/10*cos(t),k/10*sin(t)),(t,0,2*pi))
    g=g+g1
show(g)
```



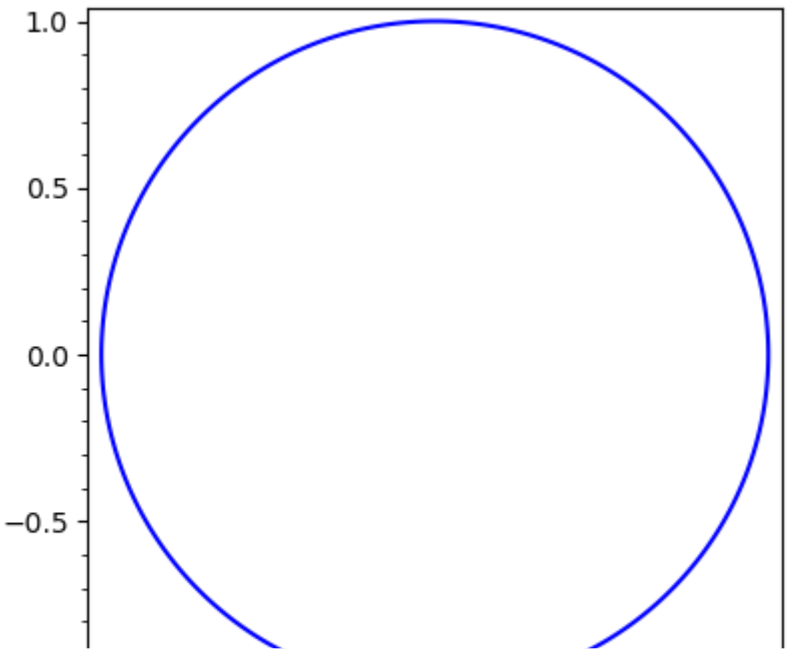
Curbe in forma implicita

Pentru a reprezenta grafic o curba in forma implicita trebuie folosita comanda

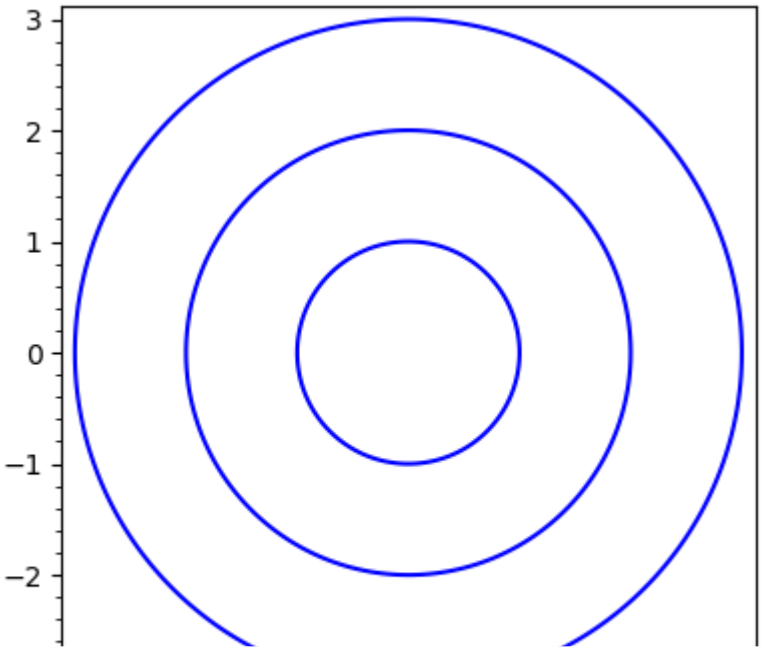
implicit_plot(f(x, y), (x, a, b), (y, c, d))

In [47]:
```
x,y=var('x,y')
f(x,y)=x^2+y^2
implicit_plot(f(x,y)==1,(x,-1,1),(y,-1,1))
```

Out[47]:



In [48]:
```
g1=implicit_plot(f(x,y)==1,(x,-1,1),(y,-1,1))
g2=implicit_plot(f(x,y)==4,(x,-2,2),(y,-2,2))
g3=implicit_plot(f(x,y)==9,(x,-3,3),(y,-3,3))
show(g1+g2+g3)
```



In [ ]:
```

```

In [ ]: