In [1]:
```
F=(x^4-4)/(x-sqrt(2))
```

In [2]:
```
F
```

Out[2]:
```
(x^4 - 4)/(x - sqrt(2))
```

In [3]:
```
F.gcd(x^2-2)
```

Out[3]:
```
x^2 - 2
```

In [4]:
```
R.<x>=PolynomialRing(QQ[sqrt(2)])
```

In [5]:
```
R(F)
```

Out[5]:
```
x^3 + sqrt2*x^2 + 2*x + 2*sqrt2
```

In [6]:
```
R(F).gcd(x^2-2)
```

Out[6]:
```
x + sqrt2
```

In [7]:
```
2+2
```

Out[7]:
```
4
```

In [8]:
```
2
```

Out[8]:
```
2
```

In [9]:
```
a=exp(2*i*pi/7)
```

In [10]:
```
S=sum([a^k for k in range(10,17)])
```

In [11]:
```
simplify(S)
```

Out[11]:
```
e^(6/7*I*pi) + e^(4/7*I*pi) + e^(2/7*I*pi) + e^(-2/7*I*pi) + e^(-4/7*I*pi) + e^
(-6/7*I*pi) + 1
```

In [12]:
```
M1=matrix([[1,2,3],[4,5,6]])
```

In [13]:
```
M1.kernel()  # attention c'est un noyau a gauche !!
```

Out[13]:
```
Free module of degree 2 and rank 0 over Integer Ring
Echelon basis matrix:
[]
```

In [14]:
```
M1.right_kernel()
```

Out[14]:
```
Free module of degree 3 and rank 1 over Integer Ring
Echelon basis matrix:
[ 1 -2  1]
```

In [15]:
```
a=x+1;b=a+1;f(x)=b
```

In [16]:
```
f
```

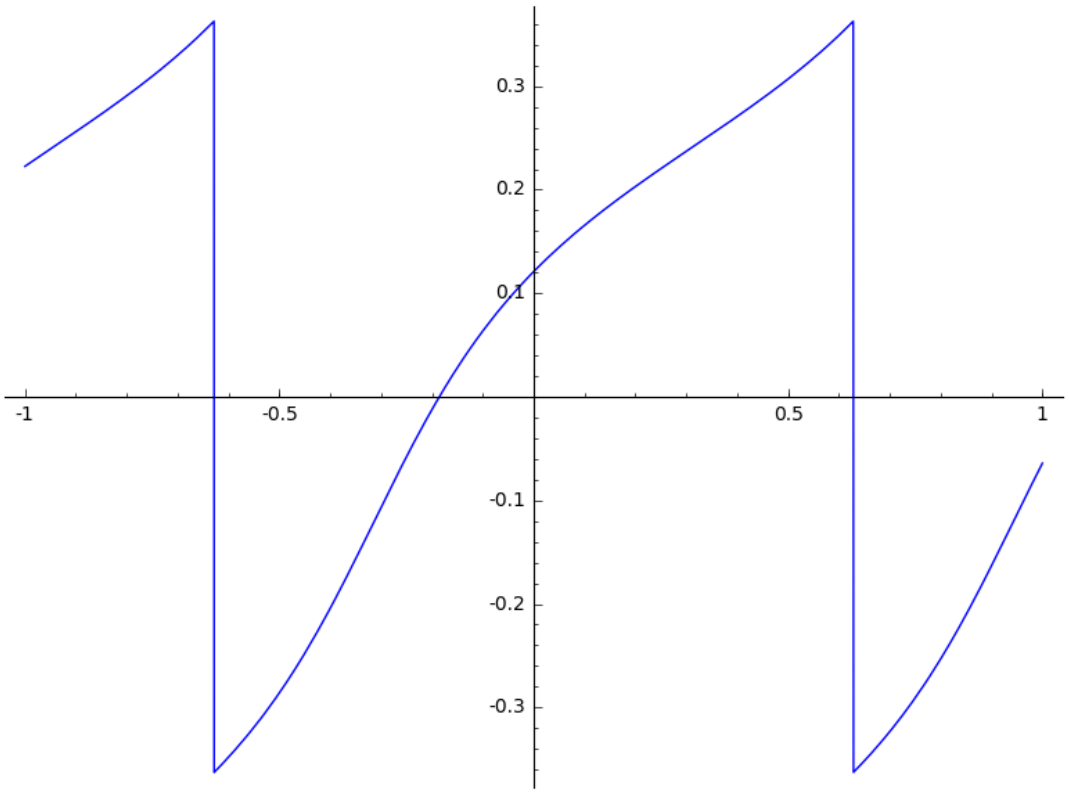
Out[16]:
```
x |--> x + 2
```

In [17]:
```
I2=(1/(2+sin(5*x))).integrate(x) ; I2
```

Out[17]:
```
2/15*sqrt(3)*arctan(1/3*sqrt(3)*(2*sin(5*x)/(cos(5*x) + 1) + 1))
```

In [18]:
```
plot(I2)  # Pb la primitive n'est que locale
```

Out[18]:



I3.plot()

In [19]:
```
M(t)=(cos(t)/sin(t)^3,sin(t)/sin(t)^3)
```

In [20]:
```
d1=parametric_plot(M(t),(t,-pi,pi))
```

In [21]:
```
d1.show(xmin=-5,xmax=5,ymin=-1,ymax=10)
```

In [22]:
```
u,x,y=var('u,x,y');
```

In [28]:
```
R.<u,x,y>=PolynomialRing(QQ)
```

In [29]:
```
(M(2*arctan(u)))[0].trig_simplify()
```
Out[29]:
```
-1/8*(u^6 + u^4 - u^2 - 1)/u^3
```

In [30]:
```
xu,yu=M(2*arctan(u))
```

In [31]:
```
xu=xu.trig_simplify();xu
```
Out[31]:
```
-1/8*(u^6 + u^4 - u^2 - 1)/u^3
```

In [32]:
```
yu=yu.trig_simplify();yu
```
Out[32]:
```
1/4*(u^4 + 2*u^2 + 1)/u^2
```

In [33]:
```
type(xu)
```
Out[33]:
```
<type 'sage.symbolic.expression.Expression'>
```

In [70]:
```
myst1=R(xu.denominator()*x-xu.numerator()).resultant(R(yu.denominator()*y-yu.num
```

In [71]:
```
myst1.unit()
```
Out[71]:
```
4096
```

In [39]:
```
R(xu.numerator())
```
Out[39]:
```
-u^6 - u^4 + u^2 + 1
```

In [41]:
```
R(xu.denominator()*x-xu.numerator())
```
Out[41]:
```
u^6 + u^4 + 8*u^3*x - u^2 - 1
```

In [75]:
```
mystere=(myst1)[0][0]
```

In [80]:
```
mystere.substitute(x=(M(t))[0],y=(M(t))[1]).simplify()
```
Out[80]:
```
-cos(t)^2/sin(t)^6 - 1/sin(t)^4 + 1/sin(t)^6
```

In [104]:
```
f(x,y)=mystere
```

In [96]:
```
diff(f(M(t)[0],M(t)[1])).simplify()
```
Out[96]:
```
0
```

In [194]:
```
l=(1,2);type(l)
```
Out[194]:
```
<type 'tuple'>
```

In [196]:
```
f(*l)
```
Out[196]:
```
3
```

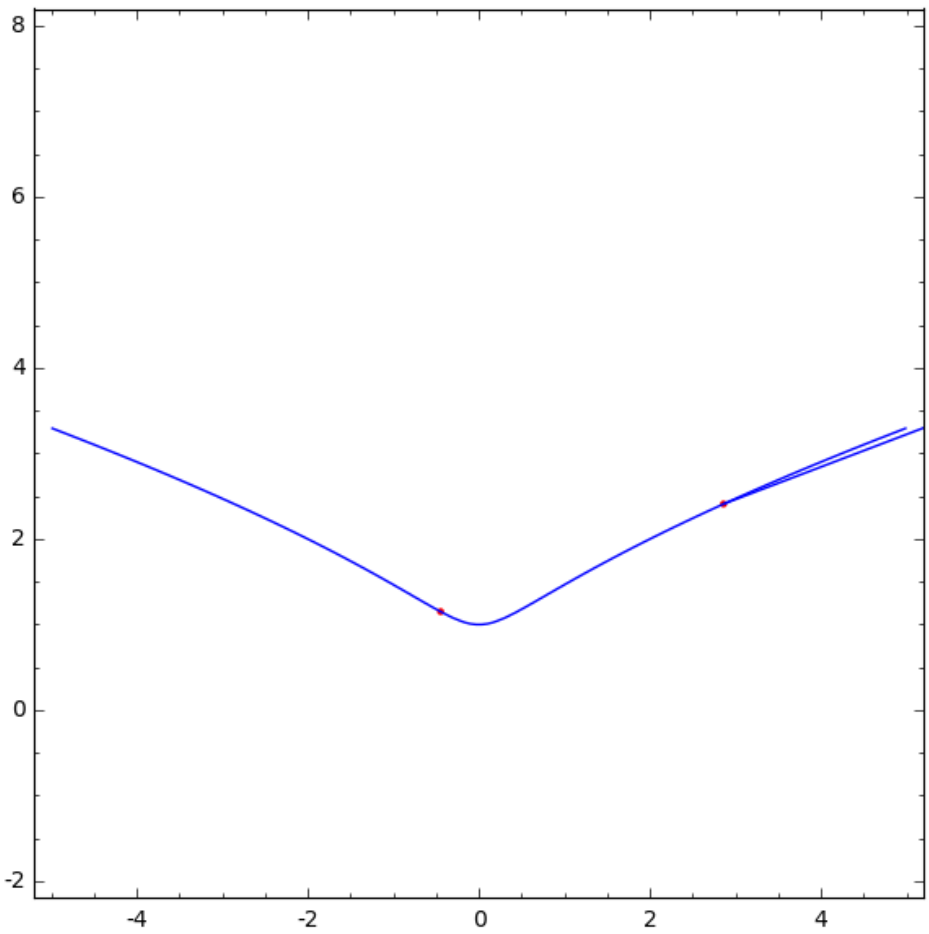
In [110]:
```
f(*M(t))
```
Out[110]:
```
-cos(t)^2/sin(t)^6 - 1/sin(t)^4 + 1/sin(t)^6
```

In [121]:
```
(f(*(M(t)))).trig_simplify()
```
Out[121]:
```
0
```

In [192]:
```
@interact(t1=0.7,t2=0.5)
def dd(t1=0.7,t2=0.9):
    D=implicit_plot(f(x,y),(x,-5,5),(y,-1,8))
    M1=point(M(t1),rgbcolor=(1,0,0))
    M2=point(M(t2),rgbcolor=(1,0,0))
    D+=line([M(t1),M(t2)],rgbcolor=(0,0,1))
    D+=M1
    D+=M2
    D+=point(M(-t1-t2),rgbcolor=(1,0,0))
    return D.show(xmin=-5,xmax=5,ymin=-2,ymax=8)
```

t1 0.70

t2 0.50



In [ ]: