m13s16_plots_integrals

An indefinite integral

```
integral(cos(x), x)
```

$$\sin(x)$$

A definite integral

```
integral(cos(x), x, 0, pi/3)
```

$$\frac{1}{2}\sqrt{3}$$

Introduce a new variable y and evaluate an iterated integral.

```
var('y')
integral(integral(x+log(y) ,y,1,4), x,4,7)
```

$$12\log(4) + \frac{81}{2}$$

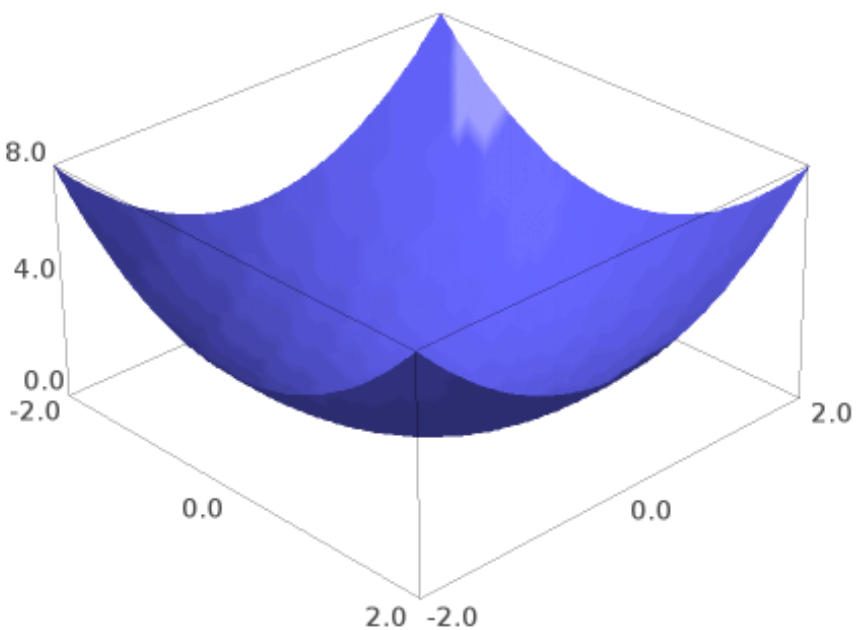
This integral is from the section on polar coordinates. Sage gives an answer, but not the one you were looking for.

```
integral(integral(exp(x^2 + y^2),y,0,sqrt(4-x^2)),x,0,2)
```

$$-\frac{1}{2} i \sqrt{\pi} \int_0^2 \operatorname{erf}\left(i \sqrt{-x^2+4}\right) e^{\left(x^2\right)} \, dx$$

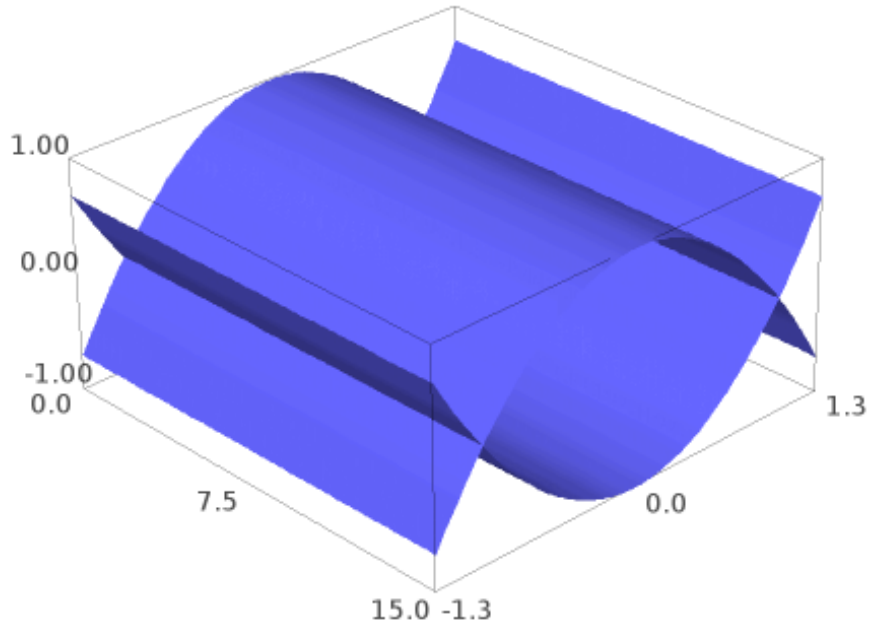
A simple 3d plot. Note the axes do not pass through the origin

```
plot3d(x^2 + y^2,(x,-2,2),(y,-2,2))
```



Draw two surfaces on the same axes.

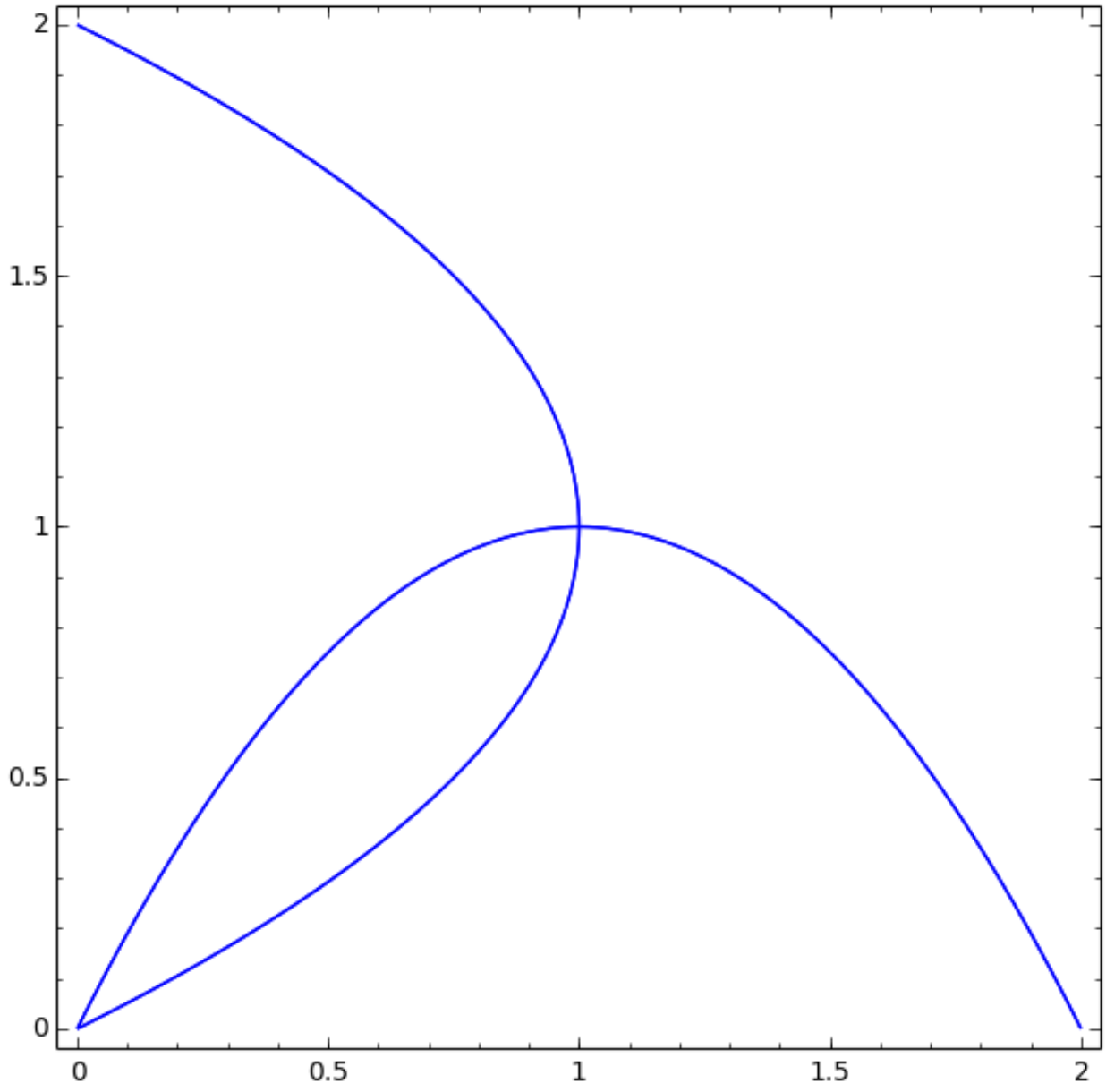
```
bottom=plot3d(y^2 -1,(x,0,15),(y,-1.3,1.3)); top=plot3d(1-y^2,(x,0,15),(y,-1.3,1.3));
show(bottom+top)
```



Implicit plot draws f(x,y) = 0

```
implicit_plot(y-2*x + x^2, (x,0,2),(y,0,2))+implicit_plot(x- 2*y + y^2, (x,0,2),
```

```
(y,0,2))
```



Can't rememer a command. Type a couple of letters and hit tab; Sage will give all completions. Don't know the syntax? Add a question mark and shift-enter

```
integral?
```

File: /usr/local/sage-6.7-x86_64-Linux/local/lib/python2.7/site-packages/sage/misc/functional.py

Type: <type 'function'>

Definition: integral(x, *args, **kwds)

Docstring:

Returns an indefinite or definite integral of an object x.

First call x.integral() and if that fails make an object and integrate it using Maxima, maple, etc, as specified by algorithm.

For symbolic expression calls `sage.calculus.calculus.integral()` - see this function for available options.

EXAMPLES:

```
sage: f = cyclotomic_polynomial(10)
sage: integral(f)
1/5*x^5 - 1/4*x^4 + 1/3*x^3 - 1/2*x^2 + x
```

```
sage: integral(sin(x),x)
-cos(x)
```

```
sage: y = var('y')
sage: integral(sin(x),y)
y*sin(x)
```

```
sage: integral(sin(x), x, 0, pi/2)
1
sage: sin(x).integral(x, 0,pi/2)
1
sage: integral(exp(-x), (x, 1, oo))
e^(-1)
```

Numerical approximation:

```
sage: h = integral(tan(x)/x, (x, 1, pi/3)); h
integrate(tan(x)/x, x, 1, 1/3*pi)
sage: h.n()
0.07571599101...
```

Specific algorithm can be used for integration:

```
sage: integral(sin(x)^2, x, algorithm='maxima')
1/2*x - 1/4*sin(2*x)
sage: integral(sin(x)^2, x, algorithm='sympy')
-1/2*cos(x)*sin(x) + 1/2*x
```

TESTS:

A symbolic integral from trac ticket #11445 that was incorrect in earlier versions of Maxima:

```
sage: f = abs(x - 1) + abs(x + 1) - 2*abs(x)
sage: integrate(f, (x, -Infinity, Infinity))
2
```

Another symbolic integral, from trac ticket #11238, that used to return zero incorrectly; with Maxima 5.26.0 one gets `1/2*sqrt(pi)*e^(` expression is less pleasant, but still has the same value. Unfortunately, the computation takes a very long time with the default settings, sc

```
sage: sage.calculus.calculus.maxima('domain: real')
real
sage: f = exp(-x) * sinh(sqrt(x))
sage: t = integrate(f, x, 0, Infinity); t            # long time
1/4*sqrt(pi)*(erf(1) - 1)*e^(1/4) - 1/4*(sqrt(pi)*(erf(1) - 1) - sqrt(pi) + 2*e^(-1) - 2)*e^(1/4) +
sage: t.canonicalize_radical()  # long time
1/2*sqrt(pi)*e^(1/4)
sage: sage.calculus.calculus.maxima('domain: complex')
complex
```

An integral which used to return -1 before maxima 5.28. See trac ticket #12842:

```
sage: f = e^(-2*x)/sqrt(1-e^(-2*x))
sage: integrate(f, x, 0, infinity)
1
```

This integral would cause a stack overflow in earlier versions of Maxima, crashing sage. See trac ticket #12377. We don't care about the re

```
sage: y = (x^2)*exp(x) / (1 + exp(x))^2
sage: _ = integrate(y, x, -1000, 1000)
```

```
derivative?
```

File: /usr/local/sage-6.7-x86_64-Linux/local/lib/python2.7/site-packages/sage/calculus/functional.py

Type: <type 'function'>

Definition: derivative(f, *args, **kwds)

Docstring:

The derivative of f.

Repeated differentiation is supported by the syntax given in the examples below.

ALIAS: diff

EXAMPLES: We differentiate a callable symbolic function:

```
sage: f(x,y) = x*y + sin(x^2) + e^(-x)
sage: f
(x, y) |--> x*y + e^(-x) + sin(x^2)
sage: derivative(f, x)
(x, y) |--> 2*x*cos(x^2) + y - e^(-x)
sage: derivative(f, y)
(x, y) |--> x
```

We differentiate a polynomial:

```
sage: t = polygen(QQ, 't')
sage: f = (1-t)^5; f
-t^5 + 5*t^4 - 10*t^3 + 10*t^2 - 5*t + 1
sage: derivative(f)
-5*t^4 + 20*t^3 - 30*t^2 + 20*t - 5
sage: derivative(f, t)
-5*t^4 + 20*t^3 - 30*t^2 + 20*t - 5
sage: derivative(f, t, t)
-20*t^3 + 60*t^2 - 60*t + 20
sage: derivative(f, t, 2)
-20*t^3 + 60*t^2 - 60*t + 20
sage: derivative(f, 2)
-20*t^3 + 60*t^2 - 60*t + 20
```

We differentiate a symbolic expression:

```
sage: var('a x')
(a, x)
sage: f = exp(sin(a - x^2))/x
sage: derivative(f, x)
-2*cos(-x^2 + a)*e^(sin(-x^2 + a)) - e^(sin(-x^2 + a))/x^2
sage: derivative(f, a)
cos(-x^2 + a)*e^(sin(-x^2 + a))/x
```

Syntax for repeated differentiation:

```
sage: R.<u, v> = PolynomialRing(QQ)
sage: f = u^4*v^5
sage: derivative(f, u)
4*u^3*v^5
sage: f.derivative(u)   # can always use method notation too
4*u^3*v^5
```

```
sage: derivative(f, u, u)
12*u^2*v^5
sage: derivative(f, u, u, u)
24*u*v^5
sage: derivative(f, u, 3)
24*u*v^5
```

```
sage: derivative(f, u, v)
20*u^3*v^4
sage: derivative(f, u, 2, v)
60*u^2*v^4
sage: derivative(f, u, v, 2)
80*u^3*v^3
sage: derivative(f, [u, v, v])
80*u^3*v^3
```