Siena Lecture - Calculus, Plotting & Interact

Calculus, plotting & interact

Symbolic Variables

To define a new variable, use the **var** command.

```
var('y t')
```
```
    (y, t)
```
```
y, t
```
```
    (y, t)
```
```
type(y)
```
```
    <class 'sage.calculus.calculus.SymbolicVariable'>
```

By default, **x** is defined to be a variable when Sage starts.

```
x
```
```
    x
```
```
type(x)
```
```
    <class 'sage.calculus.calculus.SymbolicVariable'>
```

We can form *symbolic arithmetic expressions* using variables. For example:

```
x^2 + 2*sqrt(y)
```
```
    2*sqrt(y) + x^2
```
```
type(x^2 + 2*sqrt(y))
```
```
    <class 'sage.calculus.calculus.SymbolicArithmetic'>
```

Symbolic Functions

To define a *symbolic function*, use the **function** command.

```
function('f, g')
```
```
    (f, g)
```

```
type(f)
```

```
<class 'sage.calculus.calculus.SymbolicFunction'>
```

```
f(x) + g(x,y)
```

```
g(x, y) + f(x)
```

```
type(f(x) + g(x,y))
```

```
<class 'sage.calculus.calculus.SymbolicArithmetic'>
```

```
diff(f(x)*g(x),x)
```

```
f(x)*diff(g(x), x, 1) + g(x)*diff(f(x), x, 1)
```

```
diff(f(x)/g(x),x)
```

```
diff(f(x), x, 1)/g(x) - f(x)*diff(g(x), x, 1)/g(x)^2
```

Symbolic expressions

```
f(x,y) = e^y * sin(x)
```

```
f(y=3)
```

```
e^3*sin(x)
```

```
f(x=3,y=5)
```

```
e^5*sin(3)
```

```
f(x=y, y=g(t))
```

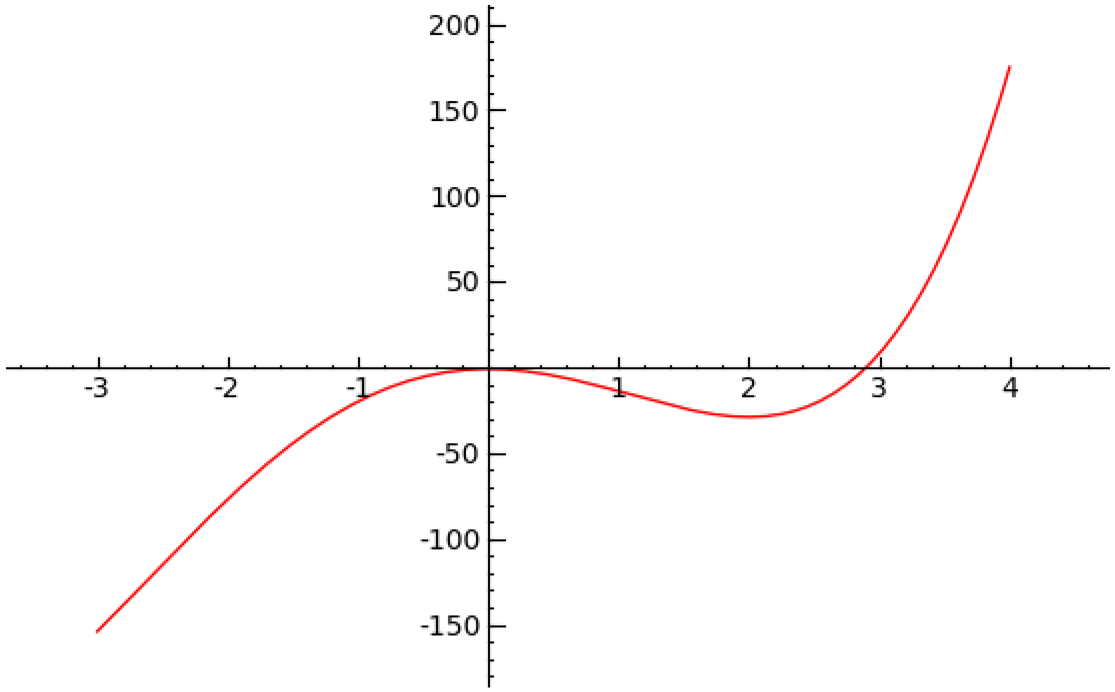
```
e^g(t)*sin(y)
```

Plotting functions

To plot a function, use the **plot** command. It has the same syntax as *Mathematica*'s plot command.

```
f(x) = x^4 + 3*x^3 - 17*x^2- 1
```

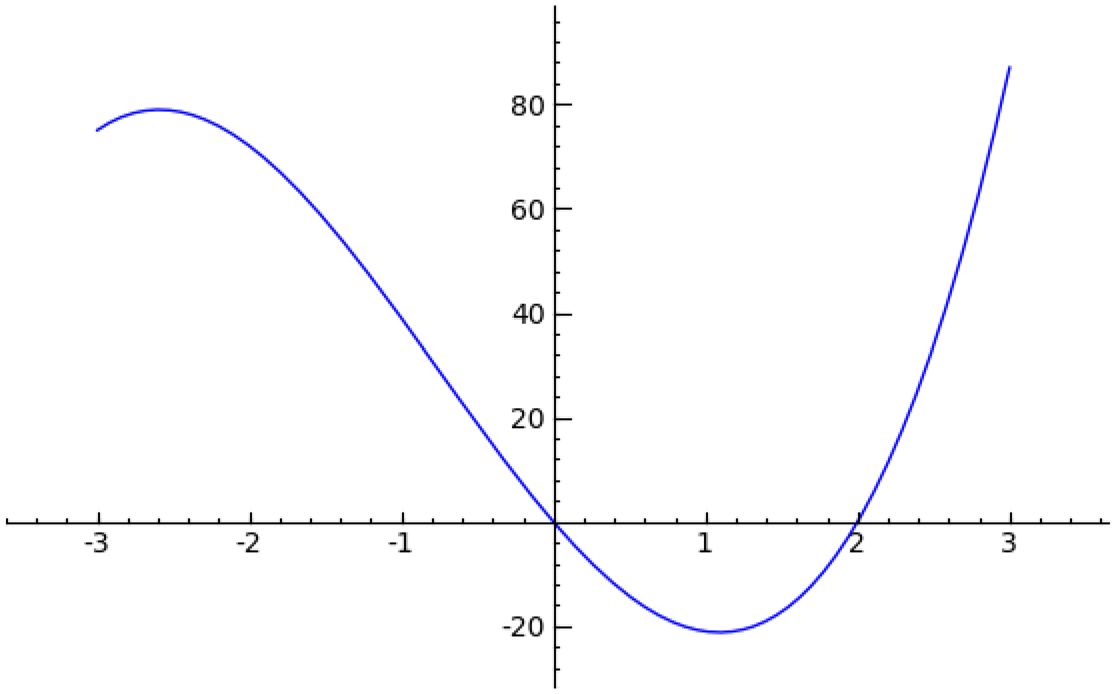
```
plot(f(x), -3, 4, color='red')
```



```
g(x) = diff(f, x)
g(x)
```

```
    4*x^3 + 9*x^2 - 34*x
```

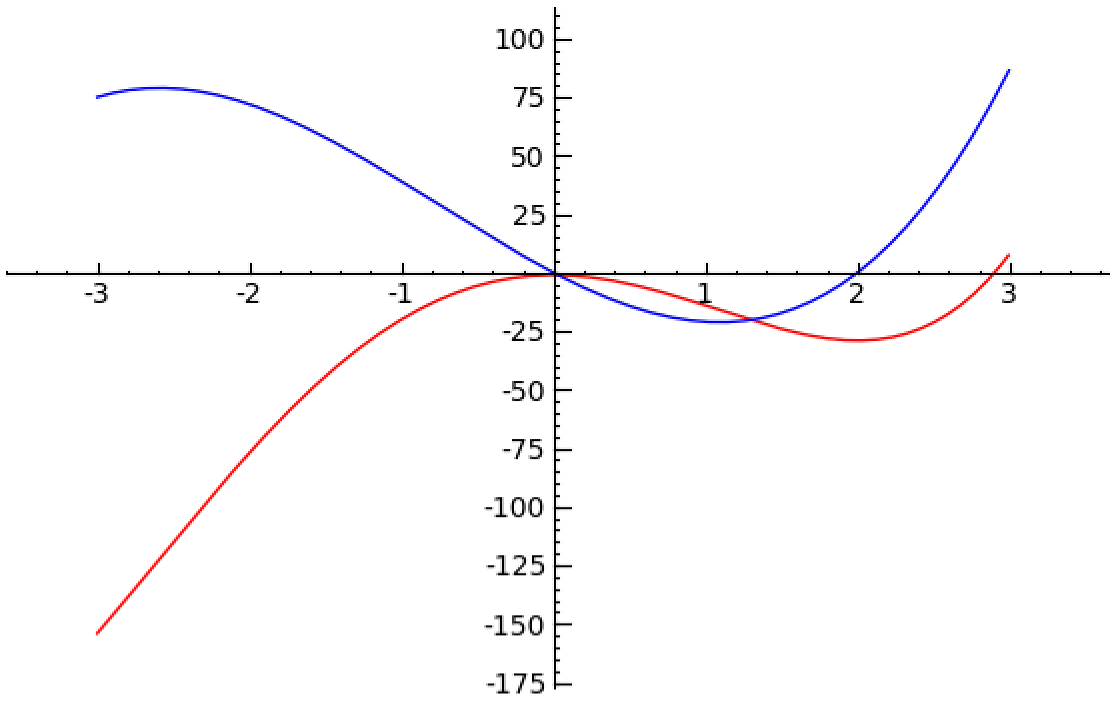
```
plot(g(x), -3, 3, color='blue')
```



Combining plots

To combine to plots objects together into one plot, *add* (+) the plots together!

```
P1 = plot(f(x), -3, 3, color='red')
P2 = plot(g(x), -3, 3, color='blue')
P1 + P2
```



Interactive Mathematics

Here is a Sage function that plots $f(x) = x^2 + sin(4x)$ and the Taylor series of f up to the given **order**.

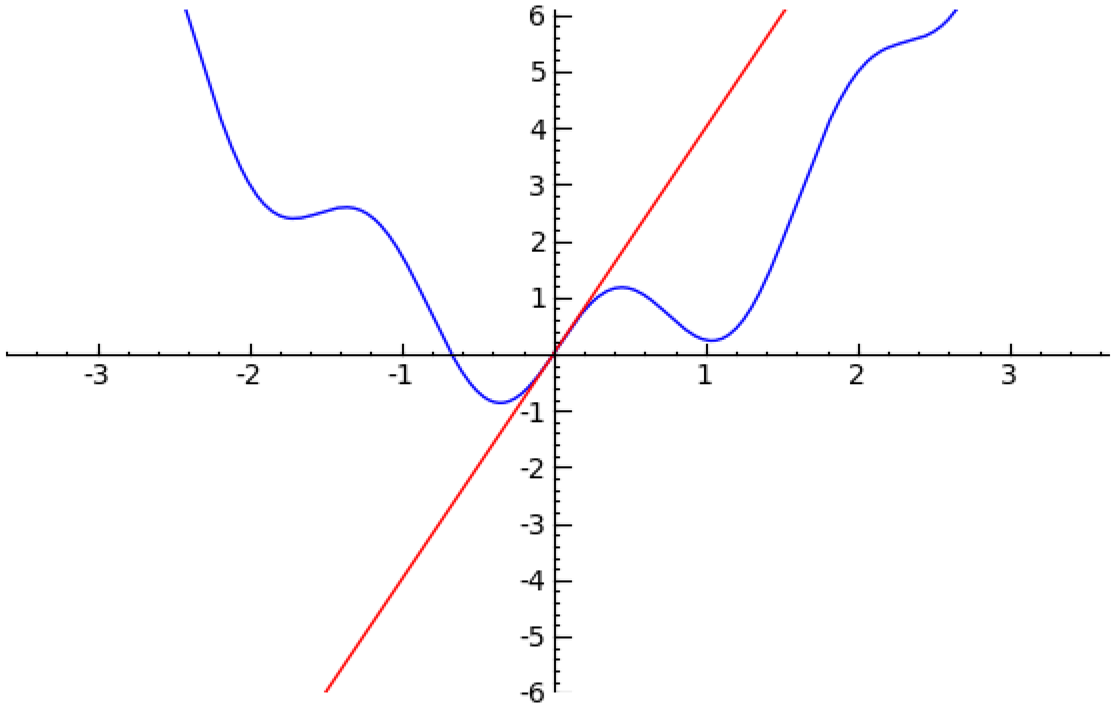
```
def taylor_plot(order):
   var('x')

   f(x) = x^2 + sin(4*x)
   f_plot = plot(f, -3, 3)
```

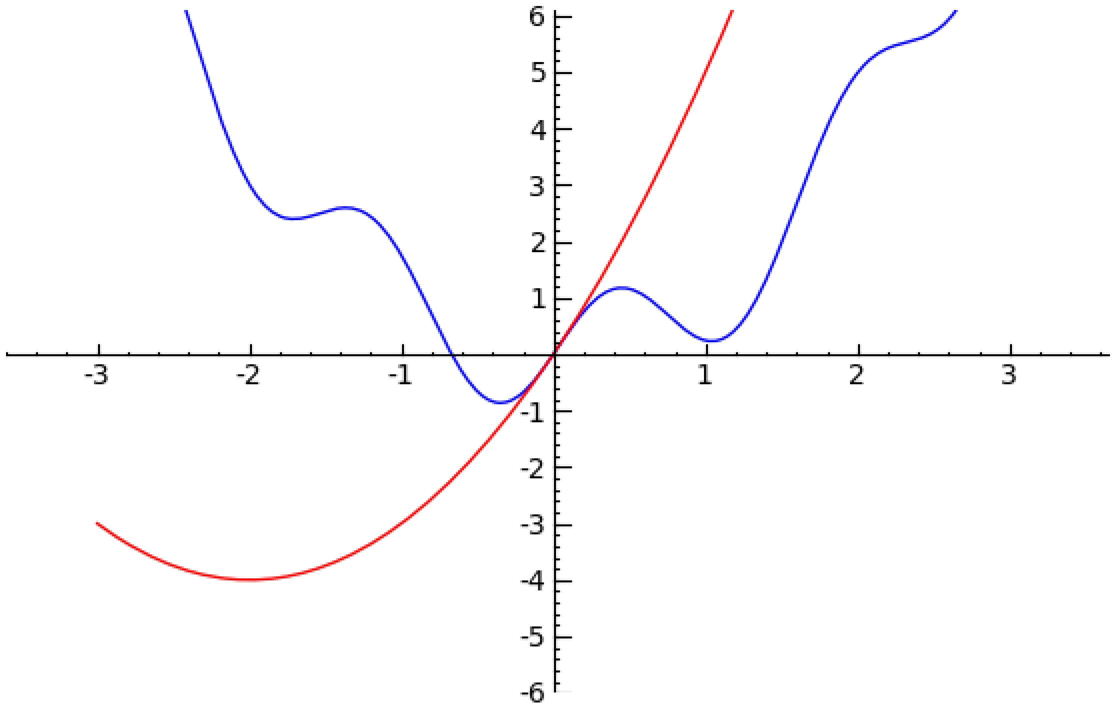
```
    g = f.taylor(x, 0, order)
    g_plot = plot(g, -3, 3, color='red')

    show(f_plot + g_plot, ymin = -5, ymax = 5)
```

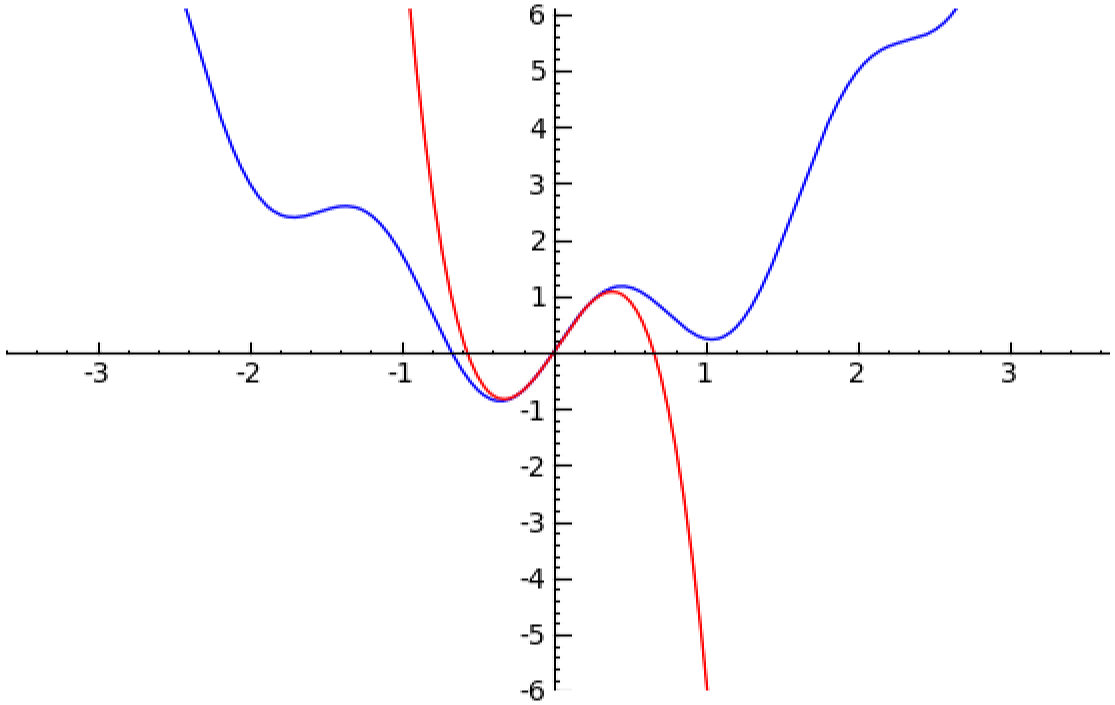
```
taylor_plot(1)
```



```
taylor_plot(2)
```



```
taylor_plot(3)
```



@interact

In order to easily *interact* with this function, we can use the **@interact** decorator. Just place **@interact** before the function and define a default for the arguments.

```
@interact
def taylor_plot(order=2):
   var('x')

   f(x) = x^2 + sin(4*x)
   f_plot = plot(f, -3, 3)

   g = f.taylor(x, 0, order)
   g_plot = plot(g, -3, 3, color='red')

   show(f_plot + g_plot, ymin = -5, ymax = 5)
```

@interact supports several very nice controllers to change the arguments. In the example below, we turn the argument **order** into a slider.

```
@interact
def taylor_plot(order=slider(0, 20, 2, default=2)):
   var('x')

   f(x) = x^2 + sin(4*x)
   f_plot = plot(f, -3, 3)

   g = f.taylor(x, 0, order)
   g_plot = plot(g, -3, 3, color='red')

   show(f_plot + g_plot, ymin = -5, ymax = 5)
```

```
@interact
def taylor_plot(order=slider(0, 20, 2, default=2),
x0=slider(-3,3,1,0), xrange=range_slider(-3,3,1,default=(-3,3))):
   var('x')

   xmin, xmax = xrange

   f(x) = x^2 + sin(4*x)
   f_plot = plot(f, xmin, xmax)
```

```
g = f.taylor(x, x0, order)
g_plot = plot(g, xmin, xmax, color='red')

show(f_plot + g_plot, ymin = -5, ymax = 5)
```