Bentley University GR526 in Python using SymPy

Content extracted from the How to Data Website

This PDF generated on 27 January 2023

Contents

GR526 is a graduate course at Bentley University that gives an overview of Calculus from a computational viewpoint, for students who plan to study quantitative finance. The description from the course catalog can be found here.

Topics include limits, derivatives, integrals, differential equations, implicit differentiation, Taylor series, and continuous probability. By-hand computation is minimized and the use of a computer algebra system is required, such as Maxima or SymPy.

Content last modified on 07 June 2021.

How to do basic mathematical computations

Description

How do we write the most common mathematical operations in a given piece of software? For example, how do we write multiplication, or exponentiation, or logarithms, in Python vs. R vs. Excel, and so on?

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                    # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

Mathematical notation	Python code	Requires SymPy?
$x + y$	`x+y`	no
$x - y$	`x-y`	no
xy	`x*y`	no
$\frac{x}{y}$	`x/y`	no
$\left\lfloor \frac{x}{y} \right\rfloor$	`x//y`	no
remainder of $x \div y$	`x%y`	no
x^y	`x**y`	no
$\lvert x \rvert$	`abs(x)`	no
$\ln x$	`log(x)`	yes
$\log_a b$	`log(b,a)`	yes
e^x	`E`	yes
π	`pi`	yes
$\sin x$	`sin(x)`	yes
$\sin^{-1} x$	`asin(x)`	yes
$\sqrt{x}$	`sqrt(x)`	yes

Other trigonometric functions are also available besides just `sin`, including `cos`, `tan`, etc.

Note that SymPy gives precise answers to mathematical queries, which may not be what you want.

```
sqrt(2)
```

$\sqrt{2}$

If you want a decimal approximation instead, you can use the `N` function.

```
N(sqrt(2))
```

1.4142135623731

Or you can use the `evalf` function.

```
sqrt(2).evalf()
```

1.4142135623731

By contrast, if you need an exact rational number when Python gives you an approximation, you can use the `Rational` function to build one. Note the differences below:

```
1/3
```

0.333333333333333

```
Rational(1,3)
```

$$\frac{1}{3}$$

Content last modified on 14 July 2021.

See a problem? Tell us or edit the source.

How to create symbolic variables

Description

The word "variable" does not mean the same thing in mathematics as it does in computer programming. In mathematics, we often use it to mean an unknown for which we might solve; but in programming, variables typically have known values.

If we want to do symbolic mathematics in a software package, how can we tell the computer that we want to use variables in the mathematical sense, as symbols whose value may be unknown?

Related tasks:

- How to substitute a value for a symbolic variable

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                  # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

You can define any number of variables as follows. Here we define x, y, and z.

```
var( 'x y z' )
```

$(x,\ y,\ z)$

You can tell that they are variables, because when you ask Python to print them out, it does not print a value (such as a number) but rather just the symbol itself.

```
x
```

x

And when you use a symbol inside a larger formula, it doesn't attempt to compute a result, but stores the entire formula symbolically.

```
formula = sqrt(x) + 5
formula
```

$\sqrt{x} + 5$

Content last modified on 01 June 2021.

See a problem? Tell us or edit the source.

How to substitute a value for a symbolic variable

Description

If we've defined a symbolic variable and used it in a formula, how can we substitute a value in for it, to evaluate the formula? This is often informally called "plugging in" a value.

Related tasks:

- How to create symbolic variables

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                  # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

Let's assume we've defined a variable and created a formula, as covered in how to create symbolic variables.

```
var( 'x' )
formula = x**2 + x
formula
```

$x^2 + x$

We can substitute a value for x using the `subs` function. You provide the variable and the value to substitute.

```
formula.subs( x, 8 )  # computes 8**2 + 8
```

72

If you had to substitute values for multiple variables, you can use multiple `subs` calls or you can pass a dictionary to `subs`.

```
var( 'y' )
formula = x/2 + y/3
formula
```

$\frac{x}{2} + \frac{y}{3}$

```
formula.subs( x, 10 ).subs( y, 6 )
```

7

```
formula.subs( { x: 10, y: 6 } )
```

7

Content last modified on 01 June 2021.

See a problem? Tell us or edit the source.

How to compute the domain of a function

Description

Given a mathematical function $f(x)$, we often want to know the set of x values for which the function is defined. That set is called its domain. How can we compute the domain of $f(x)$ using mathematical software?

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                    # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

We also need to import another tool that SymPy doesn't pull in by default.

```
from sympy.calculus.util import continuous_domain
```

We can then ask about a function's domain. We provide the function, the variable we're asking about, and the set of numbers we're working inside of. For a simple one-variable function, we're typically working in just the real numbers.

```
var( 'x' )
formula = 1 / ( x + 1 )
continuous_domain( formula, x, S.Reals )
```

$(-\infty, -1) \cup (-1, \infty)$

It's sometimes easier to instead ask where the function is *not* defined. We can just ask for the complement of the domain.

```
domain = continuous_domain( formula, x, S.Reals )
Complement( S.Reals, domain )
```

$\{-1\}$

The function is *undefined* only at $x = -1$.

Content last modified on 13 July 2021.

See a problem? Tell us or edit the source.

How to graph mathematical functions

Description

Assume we have a mathematical formula and we would like to plot a graph of it using the standard Cartesian coordinate system.

Related tasks:

- How to graph curves that are not functions
- How to graph mathematical sequences
- How to graph a two-variable function as a surface

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                    # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

You can write a formula and plot it in just a few lines of code.

```
var( 'x' )
formula = x**2 - 5*x + 9
plot( formula )
```

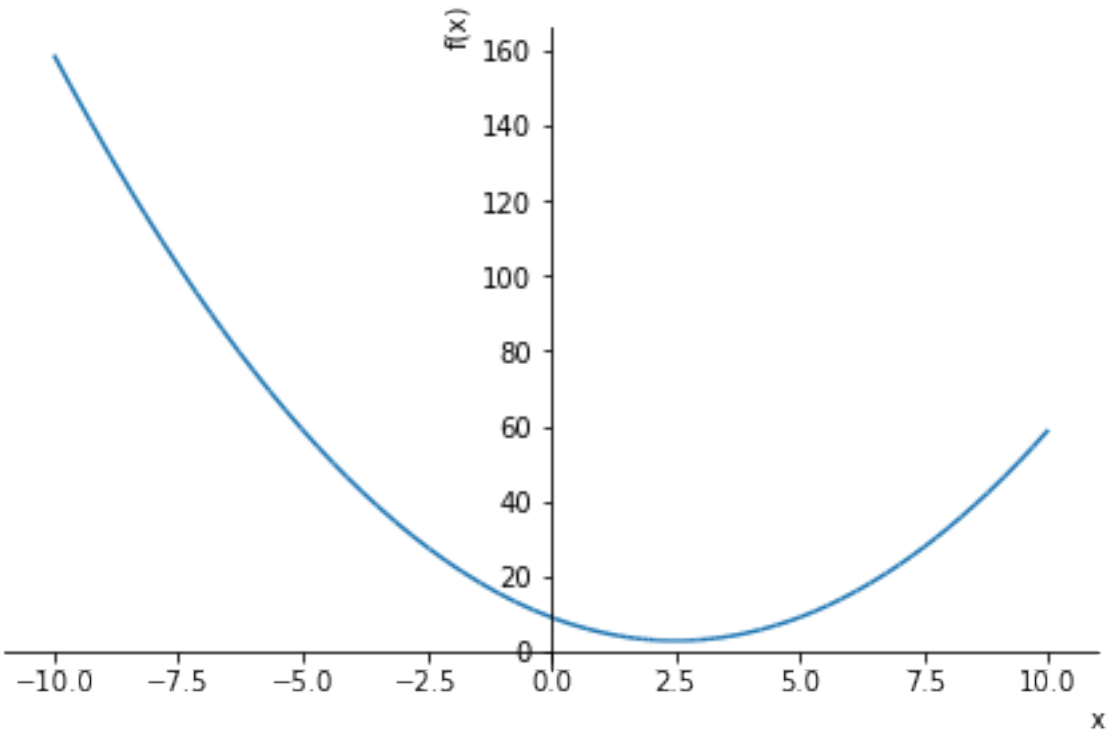

Figure 1: png

```
<sympy.plotting.plot.Plot at 0x7f42e07651c0>
```

If you want to elimiate the extra bit of text after the graph, just assign the plot to a variable, as in `p = plot( formula )`.

You can also plot more than one function on the same graph.

```
formula2 = 10*sin(x) + 20
plot( formula, formula2 )
```

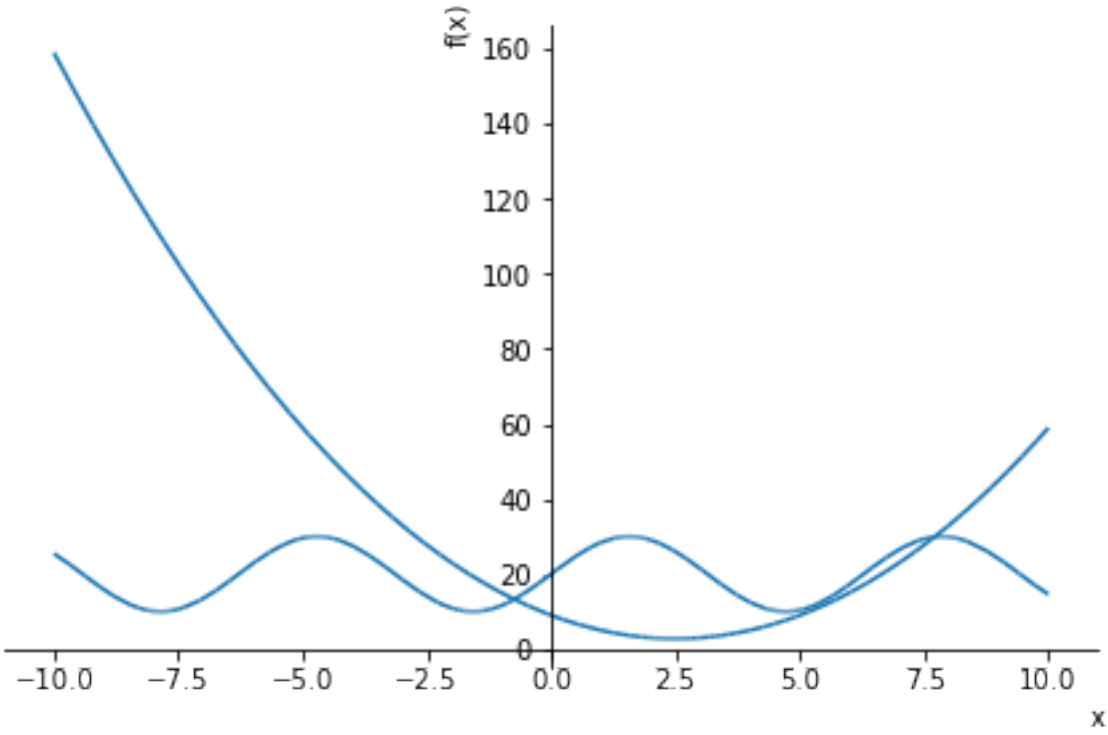

Figure 2: png

```
<sympy.plotting.plot.Plot at 0x7f42f06383d0>
```

Content last modified on 01 June 2021.

See a problem? Tell us or edit the source.

How to graph curves that are not functions

Description

Assume we have an equation in which y cannot be isolated as a function of x. (The standard example is the formula for the unit circle, $x^2 + y^2 = 1$.) We would still like to be able to use software to plot such curves. How?

Related tasks:

- How to graph mathematical functions
- How to do implicit differentiation

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                    # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

Let's consider the example of the unit circle, $x^2 + y^2 = 1$.

To plot it, SymPy first expects us to move everything to the left-hand side of the equation, so in this case, we would have $x^2 + y^2 - 1 = 0$.

We then use that left hand side to represent the equation as a single formula, and we can plot it with SymPy's `plot_implicit` function.

```
var( 'x y' )
formula = x**2 + y**2 - 1   # to represent x^2+y^2=1
plot_implicit( formula )
```

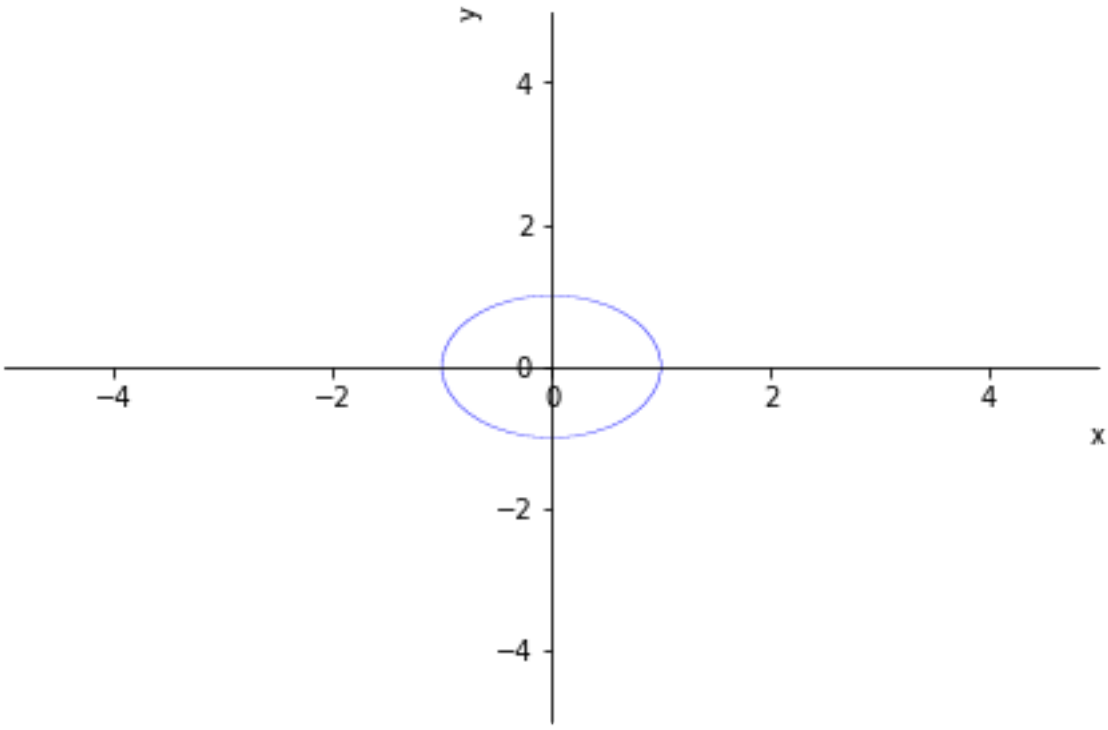

Figure 3: png

```
<sympy.plotting.plot.Plot at 0x7f10dbd7d130>
```

Content last modified on 01 June 2021.

See a problem? Tell us or edit the source.

How to write a piecewise-defined function

Description

In mathematics, we use the following notation for a "piecewise-defined" function.

$$f(x) = \begin{cases} x^2 & \text{if } x > 2 \\ 1 + x & \text{if } x \leq 2 \end{cases}$$

This means that for all x values larger than 2, $f(x) = x^2$, but for x values less than or equal to 2, $f(x) = 1+x$.

How can we express this in mathematical software?

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                    # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

SymPy has support for piecewise functions built in, using `Piecewise`. The function above would be written as follows.

```
var( 'x' )
formula = Piecewise( (x**2, x>2), (1+x, x<=2) )
formula
```

$$\begin{cases} x^2 & \text{for } x > 2 \\ x + 1 & \text{otherwise} \end{cases}$$

We can test to be sure the function works correctly by plugging in a few x values and ensuring the correct y values result. Here we're using the method from how to substitute a value for a symbolic variable.

```
formula.subs(x,1), formula.subs(x,2), formula.subs(x,3)
```

$(2,\ 3,\ 9)$

For $x = 1$ we got $1 + 1 = 2$. For $x = 2$ we got $2 + 1 = 3$. For $x = 3$, we got $3^2 = 9$.

Content last modified on 01 June 2021.

See a problem? Tell us or edit the source.

How to graph a two-variable function as a surface

Description

Assume we have a mathematical formula in the variables x and y and we would like to plot a graph of it using a 3D coordinate system.

Related tasks:

- How to graph mathematical functions
- How to graph mathematical sequences

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                   # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

First, we need a two-variable function that we wish to plot.

```
var( 'x y' )
formula = sin( x**2 + y**2 )
formula
```

$\sin\left(x^2 + y^2\right)$

You can use `plot3d`, but you have to import it specifically, because it is not imported by default with the rest of SymPy.

```
from sympy.plotting.plot import plot3d
plot3d( formula )
```

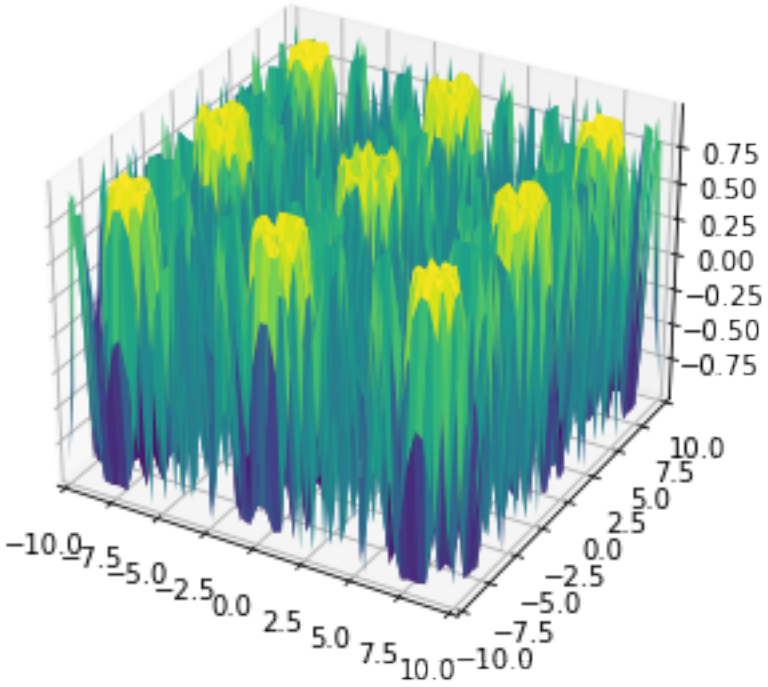

Figure 4: png

```
<sympy.plotting.plot.Plot at 0x7f76ca409280>
```

Specify the minimum and maximum values for both x and y as follows. In this example, I keep $-\pi \leq x \leq \pi$ and $-\frac{\pi}{2} \leq y \leq \frac{\pi}{2}$.

```
plot3d( formula, (x,-pi,pi), (y,-pi/2,pi/2) )
```

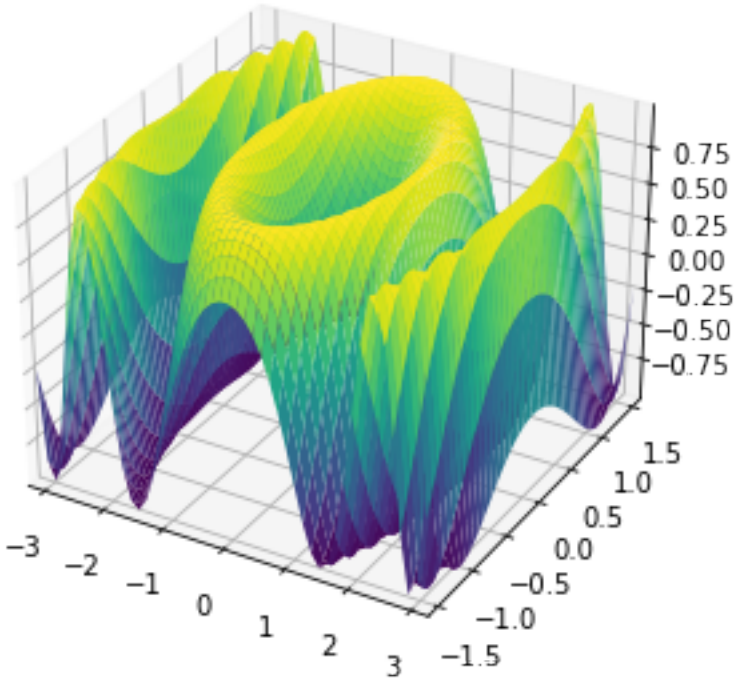

Figure 5: png

```
<sympy.plotting.plot.Plot at 0x7f763acebd30>
```

Content last modified on 01 June 2021.

See a problem? Tell us or edit the source.

How to write symbolic equations

Description

In programming, when we write `a=b`, the computer interprets it as an instruction, to change the value of `a` to `b`. But in mathematics, $a = b$ is a statement that a and b are equal; it's often a starting point for algebraic work. How can we write a mathematical equation using software?

Related tasks:

- How to solve symbolic equations
- How to isolate one variable in an equation

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                    # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

Let's say we want to write the equation $x^2 + y^2 = 2$. We must first define x and y as mathematical variables, then use SymPy's `Eq` function to build an equation. This helps SymPy distinguish a mathematical equation from a Python assignment statement.

```
var( 'x y' )
Eq( x**2 + y**2, 2 )    # Two parameters: left and right sides of equation
```

$x^2 + y^2 = 2$

You can make a system of equations just by placing several equations in a Python list.

```
system = [
    Eq( x + 2*y, 1 ),
    Eq( x - 9*y, 5 )
]
system
```

$[x + 2y = 1,\ x - 9y = 5]$

Content last modified on 10 September 2021.

See a problem? Tell us or edit the source.

How to solve symbolic equations

Description

Once we've expressed an equation or system of equations using the technique from how to write symbolic equations, we often want the software to solve the equation or system of equations for us.

Related tasks:

- How to isolate one variable in an equation

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                  # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

If your equation has just one variable, simply call `solve` on it. Note that you may get a list of more than one solution.

```
var( 'x' )
equation = Eq( x**2 + 3*x, -x + 9 )
solve( equation )
```

$$\left[-2 + \sqrt{13},\ -\sqrt{13} - 2\right]$$

Sometimes you get no solutions, which is shown as a Python empty list.

```
solve( Eq( x+1, x+2 ) )
```

[]

Sometimes the answers include complex numbers.

```
solve( Eq( x**3, -1 ) )
```

$$\left[-1,\ \frac{1}{2} - \frac{\sqrt{3}i}{2},\ \frac{1}{2} + \frac{\sqrt{3}i}{2}\right]$$

To restrict the solution to the real numbers, use `solveset` instead, and specify the real numbers as the domain.

```
solveset( Eq( x**3, -1 ), domain=S.Reals )
```

$\{-1\}$

You can solve systems of equations by calling `solve` on them.

```
var( 'x y' )
system = [
    Eq( x + 2*y, 1 ),
    Eq( x - 9*y, 5 )
]
solve( system )
```

$$\left\{x : \frac{19}{11},\ y : -\frac{4}{11}\right\}$$

Content last modified on 01 June 2021.

See a problem? Tell us or edit the source.

How to isolate one variable in an equation

Description

Once we've expressed an equation or system of equations using the technique from how to write symbolic equations, we often want the software to isolate one variable in terms of all the others.

Related tasks:

- How to solve symbolic equations

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                  # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

Let's create an equation with many variables.

```
var('P V n R T')
ideal_gas_law = Eq( P*V, n*R*T )
ideal_gas_law
```

$PV = RTn$

To isolate one variable, call the `solve` function, and pass that variable as the second argument.

```
solve( ideal_gas_law, R )
```

$$\left[\frac{PV}{Tn}\right]$$

The brackets surround a list of all solutions—in this case, just one. That solution is that $R = \frac{PV}{Tn}$.

Content last modified on 01 June 2021.

See a problem? Tell us or edit the source.

How to compute the limit of a function

Description

In mathematics, we write

$$\lim_{x \to a} f(x)$$

to refer to the value that f approaches as x gets close to a, called "the limit of $f(x)$ as x approaches a."

How can we use software to compute such limits?

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                    # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

Here we define a simple mathematical formula, $\frac{x^2-x-2}{x-2}$, and compute the limit as x approaches 2. We use SymPy's built-in `limit` function, which takes the formula $f(x)$, the variable x, and the value a.

```
var( 'x' )
formula = ( x**2 - x - 2 ) / ( x - 2 )
limit( formula, x, 2 )
```

3

You can also compute one-sided limits. For instance, the limit of $\frac{|x|}{x}$ is 1 as x approaches 0 from the right, but it is -1 as x approaches 0 from the left.

```
limit( abs(x)/x, x, 0, "-" )  # "-" means from the left
```

-1

```
limit( abs(x)/x, x, 0, "+" )  # "+" means from the right
```

1

Content last modified on 01 June 2021.

See a problem? Tell us or edit the source.

How to define a mathematical sequence

Description

In mathematics, a sequence is an infinite list of values, typically real numbers, often written $a_0, a_1, a_2, ...$, or collectively as a_n.

(Let's assume that sequences are indexed starting with index 0, at a_0, even though some definitions start with index 1, at a_1, instead.)

How can we express sequences in mathematical software?

Related tasks:

- How to define a mathematical series

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                    # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

Sequences are typically written in terms of an independent variable n, so we will tell SymPy to use n as a symbol, then define our sequence in terms of n.

We define a term of an example sequence as $a_n = \frac{1}{n+1}$, then build a sequence from that term. The code `(n,0,oo)` means that n starts counting at $n = 0$ and goes on forever (with `oo` being the SymPy notation for ∞).

```
var( 'n' )                        # use n as a symbol
a_n = 1 / ( n + 1 )               # formula for a term
seq = sequence( a_n, (n,0,oo) )  # build the sequence
seq
```

$$\left[1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots\right]$$

You can ask for specific terms in the sequence, or many terms in a row, as follows.

```
seq[20]
```

$$\frac{1}{21}$$

```
seq[:10]
```

$$\left[1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \frac{1}{7}, \frac{1}{8}, \frac{1}{9}, \frac{1}{10}\right]$$

You can compute the limit of a sequence,

$$\lim_{n \to \infty} a_n.$$

```
limit( a_n, n, oo )
```

0

Content last modified on 01 June 2021.

See a problem? Tell us or edit the source.

How to graph mathematical sequences

Description

Assume we have a mathematical sequence $a_0, a_1, a_2, ...$ and we would like to plot a graph of it using the standard Cartesian coordinate system. The result will not look like a curve, because a sequence is separate points instead of a smooth curve.

Related tasks:

- How to graph mathematical functions
- How to define a mathematical sequence

Solution in Python using SymPy

How to Data does not yet contain a solution for this task in Python using SymPy.

How to define a mathematical series

Description

In mathematics, a series is a sum of values from a sequence, typically real numbers. Finite series are written as $a_0 + a_1 + \cdots + a_n$, or

$$\sum_{i=0}^{n} a_i.$$

Infinite series are written as $a_0 + a_1 + a_2 + \cdots$, or

$$\sum_{n=0}^{\infty} a_n.$$

How can we express series in mathematical software?

Related tasks:

- How to define a mathematical series

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                    # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

We define here the same sequence we defined in the task entitled how to define a mathematical sequence.

```
var( 'n' )                        # use n as a symbol
a_n = 1 / ( n + 1 )               # formula for a term
seq = sequence( a_n, (n,0,oo) )  # build the sequence
seq
```

$$\left[1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots\right]$$

We can turn it into a mathematical series by simply replacing the word `sequence` with the word `Sum`. This does not compute the answer, but just writes the series for us to view. In this case, it is an infinite series.

```
Sum( a_n, (n,0,oo) )
```

$$\sum_{n=0}^{\infty} \frac{1}{n+1}$$

You can compute the answer by appending the code `.doit()` to the above code, which asks SymPy to "do" (or evaluate) the sum.

```
Sum( a_n, (n,0,oo) ).doit()
```

$$\infty$$

In this case, the series diverges.

We can also create and evaluate finite series by replacing the `oo` with a number.

```
Sum( a_n, (n,0,10) ).doit()
```

$$\frac{83711}{27720}$$

Content last modified on 01 June 2021.

See a problem? Tell us or edit the source.

How to compute the derivative of a function

Description

Given a mathematical function $f(x)$, we write $f'(x)$ or $\frac{d}{dx}f(x)$ to represent its derivative, or the rate of change of f with respect to x. How can we compute $f'(x)$ using mathematical software?

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                    # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

In SymPy, we tend to work with formulas (that is, mathematical expressions) rather than functions (like $f(x)$). So if we wish to compute the derivative of $f(x) = 10x^2-16x+1$, we will focus on just the $10x^2-16x+1$ portion.

```
var( 'x' )
formula = 10*x**2 - 16*x + 1
formula
```

$10x^2 - 16x + 1$

We can compute its derivative by using the `diff` function.

```
diff( formula )
```

$20x - 16$

If it had been a multi-variable function, we would need to specify the variable with respect to which we wanted to compute a derivative.

```
var( 'y' )               # introduce a new variable
formula2 = x**2 - y**2   # consider the formula x^2 + y^2
diff( formula2, y )      # differentiate with respect to y
```

$-2y$

We can compute second or third derivatives by repeating the variable with respect to which we're differentiating. To do partial derivatives, use multiple variables.

```
diff( formula, x, x )    # second derivative with respect to x
```

20

```
diff( formula2, x, y )   # mixed partial derivative
```

0

Content last modified on 31 August 2022.

See a problem? Tell us or edit the source.

How to compute the Taylor series for a function

Description

Any function that has arbitrarily many derivatives at a given point can have a Taylor series computed for the function centered at that point. How can we ask symbolic mathematics software to do this for us?

Related tasks:

- How to compute the error bounds on a Taylor approximation

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                  # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

Let's define an example function whose Taylor series we'd like to compute.

```
var( 'x' )
formula = exp( 2*x + 1 )
formula
```

e^{2x+1}

Let's ask for a degree-5 Taylor series centered at $x = 2$. From the code below, you can tell that the third parameter is the center point and the fourth parameter is the degree.

```
series( formula, x, 2, 5 )
```

$$e^5 + 2\left(x-2\right)e^5 + 2\left(x-2\right)^2 e^5 + \frac{4\left(x-2\right)^3 e^5}{3} + \frac{2\left(x-2\right)^4 e^5}{3} + O\left(\left(x-2\right)^5; x \to 2\right)$$

The final term (starting with O—oh, not zero) means that there are more terms in the infinite Taylor series not shown in this finite approximation. If you want to show just the approximation, you can tell it to remove the O term.

```
series( formula, x, 2, 5 ).removeO()
```

$$\frac{2\left(x-2\right)^4 e^5}{3} + \frac{4\left(x-2\right)^3 e^5}{3} + 2\left(x-2\right)^2 e^5 + 2\left(x-2\right)e^5 + e^5$$

You can also compute individual coefficients in a Taylor series by remembering the formula for the n^{th} term in the series and applying it, as follows. The formula for a series centered on $x = a$ is $\frac{f^{(n)}(a)}{n!}$.

From the answer above, we can see that the coefficient on the $n = 3$ term is $\frac{4}{3}e^5$.

```
n = 3
a = 2
diff( formula, x, n ).subs( x, a ) / factorial( n )
```

$$\frac{4e^5}{3}$$

Content last modified on 01 June 2021.

See a problem? Tell us or edit the source.

How to compute the error bounds on a Taylor approximation

Description

A Taylor series approximation of degree n to the function $f(x)$, centered at the point $x = a$, has an error bounded by the following formula, where c ranges over all points between $x = a$ and the point $x = x_0$ at which we will be applying the approximation.

$$\frac{|x_0 - a|^{n+1}}{(n+1)!} \max |f^{(n+1)}(c)|$$

How can we compute this error bound using mathematical software?

Related tasks:

- How to compute the Taylor series for a function

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                  # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

Let's create a simple example. We'll be approximating $f(x) = \sin x$ centered at $a = 0$ with a Taylor series of degree $n = 5$. We will be applying our approximation at $x_0 = 1$. What is the error bound?

```
var( 'x' )
formula = sin(x)
a = 0
x_0 = 1
n = 5
```

We will not ask SymPy to compute the formula exactly, but will instead have it sample a large number of c values from the interval in question, and compute the maximum over those samples. (The exact solution can be too hard for SymPy to compute.)

```
# Get 1000 evenly-spaced c values:
cs = [ Min(x_0,a) + abs(x_0-a)*i/1000 for i in range(1001) ]
# Create the formula |f^(n+1)(x)|:
formula2 = abs( diff( formula, x, n+1 ) )
# Find the max of it on all the 1000 values:
m = Max( *[ formula2.subs(x,c) for c in cs ] )
# Compute the error bound:
N( abs(x_0-a)**(n+1) / factorial(n+1) * m )
```

0.00116870970112208

The error is at most 0.00116871 ....

Content last modified on 01 June 2021.

See a problem? Tell us or edit the source.

How to do implicit differentiation

Description

Assume we have an equation in which y cannot be isolated as a function of x. (The standard example is the formula for the unit circle, $x^2 + y^2 = 1$.) We would still like to be able to compute the derivative of y with respect to x.

Related tasks:

- How to graph curves that are not functions

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                  # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

Let's consider the example of the unit circle, $x^2 + y^2 = 1$.

To plot it, SymPy first expects us to move everything to the left-hand side of the equation, so in this case, we would have $x^2 + y^2 - 1 = 0$.

We then use that left hand side to represent the equation as a single formula, and computue $\frac{dy}{dx}$ using the `idiff` function (standing for "implicit differentiation").

```
var( 'x y' )
formula = x**2 + y**2 - 1   # to represent x^2+y^2=1
idiff( formula, y, x )
```

$$-\frac{x}{y}$$

So in this case, $\frac{dy}{dx} = -\frac{x}{y}$.

Content last modified on 01 June 2021.

See a problem? Tell us or edit the source.

How to find the critical numbers of a function

Description

When trying to find the maximum and minimum values of a function, one of the main techniques in calculus is to use the "critical numbers" of the function, which are the most important x values to examine to find maxima and minima. Can we find critical numbers for a single-variable function using software?

Related tasks:

- How to compute the domain of a function

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                    # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

Let's create an example function to work with.

```
var( 'x' )
formula = sqrt( x - 1 ) - x
formula
```

$-x + \sqrt{x - 1}$

Critical numbers come in two kinds. First, where is the derivative zero? Second, where is the derivative undefined but the function is defined?

Let's begin by finding where the derivative is zero. We'll use the same techniques introduced in how to write symbolic equations and how to solve symbolic equations.

```
derivative = diff( formula )
derivative
```

$-1 + \frac{1}{2\sqrt{x - 1}}$

```
solve( Eq( derivative, 0 ) )
```

$\left[\frac{5}{4}\right]$

So one critical number, where the derivative is zero, is $x = \frac{5}{4}$.

Now where is the derivative defined but the function undefined? We compute the domain of both functions and subtract them, using the techniques from how to compute the domain of a function.

```
from sympy.calculus.util import continuous_domain
f_domain = continuous_domain( formula, x, S.Reals )
deriv_domain = continuous_domain( derivative, x, S.Reals )
Complement( f_domain, deriv_domain )
```

$\{1\}$

So another critical number, where the function is defined but the derivative is not, is $x = 1$.

Thus the full set of critical numbers for this function is $\{1, \frac{5}{4}\}$.

Content last modified on 13 July 2021.

See a problem? Tell us or edit the source.

How to write and evaluate indefinite integrals

Description

The antiderivative of a function is expressed using an indefinite integral, as in

$$\int f(x)\,dx.$$

How can we write and evaluate indefinite integrals using software?

Related tasks:

- How to compute the derivative of a function
- How to write and evaluate definite integrals

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                   # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

Let's choose an example formula whose antiderivative we will compute.

```
var( 'x' )
formula = 3*sqrt(x)
formula
```

$3\sqrt{x}$

Use the `Integral` function to build a definite integral without evaluating it. The second parameter is the variable with respect to which you're integrating.

```
Integral( formula, x )
```

$$\int 3\sqrt{x}\,dx$$

Use the `integrate` function to perform the integration, showing the answer.

```
integrate( formula, x )
```

$2x^{\frac{3}{2}}$

```
integrate( formula, x ) + var('C')  # same, but with a constant of integration
```

$C + 2x^{\frac{3}{2}}$

Content last modified on 02 June 2021.

See a problem? Tell us or edit the source.

How to write and evaluate definite integrals

Description

The area under a curve can be computed using a definite integral. To compute the area above the x axis and under $f(x)$, from $x = a$ to $x = b$, we write

$$\int_a^b f(x)\,dx.$$

How can we write and evaluate definite integrals using software?

Related tasks:

- How to compute the derivative of a function
- How to write and evaluate indefinite integrals

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                    # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

Let's compute the area under $\sin x$ from $x = 0$ to $x = \pi$.

We use the same technique as in how to write and evaluate indefinite integrals, except that we add the lower and upper bounds together with x, as shown below.

```
var( 'x' )
formula = sin(x)
Integral( formula, (x,0,pi) )
```

$$\int_0^\pi \sin(x)\,dx$$

The above code just displays the definite integral. To evaluate it, use the `integrate` command.

```
integrate( formula, (x,0,pi) )
```

2

Content last modified on 02 June 2021.

See a problem? Tell us or edit the source.

How to write an ordinary differential equation

Description

Differential equations are equations that contain differentials like dy and dx, often in the form $\frac{dy}{dx}$. How can we write them using software?

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                    # load all math functions
init_printing( use_latex='mathjax' )   # use pretty math output
```

The following code tells SymPy that x is a variable and that y is a function of x. It then expresses $\frac{dy}{dx}$ as the derivative of y with respect to x.

```
var( 'x' )                 # Let x be a variable.
y = Function('y')(x)       # Literally, y is a function, named y, based on x.
dydx = Derivative( y, x )  # How to write dy/dx.
dydx                       # Let's see how SymPy displays dy/dx.
```

$$\frac{d}{dx}y(x)$$

Let's now write a very simple differential equation, $\frac{dy}{dx} = y$.

As with how to do implicit differentiation, SymPy expects us to move everything to the left hand side of the equation. In this case, that makes the equation $\frac{dy}{dx} - y = 0$, and we will use just the left-hand side to express our ODE.

```
ode = dydx - y
ode
```

$$-y(x) + \frac{d}{dx}y(x)$$

Content last modified on 02 June 2021.

See a problem? Tell us or edit the source.

How to solve an ordinary differential equation

Description

Elsewhere we've seen how to write an ordinary differential equation. Once one is written, how can we ask software to solve it? And since ODEs often come with initial conditions that impact the solution, how can we include those as well?

Solution in Python using SymPy

This answer assumes you have imported SymPy as follows.

```
from sympy import *                    # load all math functions
init_printing( use_latex='mathjax' )  # use pretty math output
```

Let's re-use here the code from how to write an ordinary differential equation, to write $\frac{dy}{dx} = y$.

```
var( 'x' )
y = Function('y')(x)
dydx = Derivative( y, x )
ode = dydx - y
ode
```

$$-y(x) + \frac{d}{dx}y(x)$$

You can solve an ODE by using the `dsolve` command.

```
solution = dsolve( ode )
solution
```

$$y(x) = C_1 e^x$$

If there are initial conditions that need to be substituted in for x and y, it is crucial to substitute for y first and then x. Let's assume we have the initial condition $(3, 5)$. We might proceed as follows.

```
with_inits = solution.subs( y, 5 ).subs( x, 3 )
with_inits
```

$$5 = C_1 e^3$$

```
solve( with_inits )
```

$$\left[\frac{5}{e^3}\right]$$

To substitute $C_1 = \frac{5}{e^3}$ into the solution, note that C_1 is written as `var('C1')`.

```
solution.subs( var('C1'), 5/E**3 )
```

$$y(x) = \frac{5e^x}{e^3}$$

Content last modified on 02 June 2021.

See a problem? Tell us or edit the source.