Chapter 10

Symbolic Computation with SymPy

Symbolic computation is an important tool for scientists, mathematicians and engineers. Some of the most well-developed systems are Mathematica and Maple. These commercial products are not free. A good alternative is SymPy, a Python library for symbolic computation. This chapter shows, by way of example, some basic capabilities of SymPy.

SymPy is well suited for the Jupyter notebook environment. The examples below are shown as they would appear in a jupyter notebook.[1]

10.1 Basic SymPy commands

Import SymPy and define variables:

In [1]:
```
import sympy as sp                    # import sympy
x,y,z,b = sp.symbols('x y z beta')  # define variables
x,y,z,b                               # print variables
```

Out [1]: (x, y, z, β)

Define functions:

In [2]:
```
f = x**2 - y**2              # define an explicit function
f                            # print f
```

Out [2]: $x^2 - y^2$

[1]In newer versions of jupyter notebook, the words “In” and “Out” that preceed the cell numbers are omitted. For clarity of presentation, “In” and “Out” are used here.

In [3]:
```
g = x*(y + z)                   # another explicit function
g                               # print g
```
Out [3]: $x(y+z)$

Factor, expand and simplify:

In [4]:
```
sp.factor(f)                    # factor f
```
Out [4]: $(x-y)(x+y)$

In [5]:
```
sp.expand(g)                    # expand g
```
Out [5]: $xy+xz$

In [6]:
```
h = f/(x - y) + g               # define h
h                               # print h
```
Out [6]: $x(y+z)+\frac{x^2-y^2}{x-y}$

In [7]:
```
sp.simplify(h)                  # simplify h
```
Out [7]: $xy+xz+x+y$

Collect, substitute and coefficients:

In [8]:
```
p = sp.expand((x + z)*(y + z)**2)  # define p
p                                  # print p
```
Out [8]: $xy^2+2xyz+xz^2+y^2z+2yz^2+z^3$

In [9]:
```
sp.collect(p,z)         # collect terms in powers of z
```
Out [9]: $xy^2+z^3+z^2(x+2y)+z(2xy+y^2)$

In [10]:
```
p.subs(x,3)             # substitute 3 in place of x
```
Out [10]: $y^2z+3y^2+2yz^2+6yz+z^3+3z^2$

In [11]:
```
p.subs(z,2*y)           # substitute 2*y in place of z
```
Out [11]: $9xy^2+18y^3$

In [12]:
```
p.subs(((x,1),(y,2)))  # substitute 1 for x and 2 for y
```
Out [12]: z^3+5z^2+8z+4

In [13]:
```
p.coeff(z,2)        # coefficient of z^2
```
Out [13]: $x + 2y$

Equations and solutions:

In [14]:
```
f = x**2 - 5*x + 6  # new definition for f
f                   # print f
```
Out [14]: $x^2 - 5x + 6$

In [15]:
```
myeqn = sp.Eq(f,0)  # create the equation f = 0
                    # Eq stands for "Equation" or "Equals"
myeqn               # print myeqn
```
Out [15]: $x^2 - 5x + 6 = 0$

In [16]:
```
solns = sp.solve(myeqn,x)  # solve myeqn for x
solns                      # print solns
```
Out [16]: $[2, 3]$

In [17]:
```
solns[0]     # print first solution (index = 0)
```
Out [17]: 2

In [18]:
```
solns[1]     # print second solution (index = 1)
```
Out [18]: 3

In [19]:
```
myeqn.subs(x,solns[0])    # check first solution
```
Out [19]: True

In [20]:
```
myeqn.subs(x,solns[1])    # check second solution
```
Out [20]: True

Systems of equations:

In [21]:
```
g = 3*x + 2*y       # define function g
eq1 = sp.Eq(g,7)    # define equation g = 7
eq1                 # print eq1
```
Out [21]: $3x + 2y = 7$

In [22]:
```
h = 2*x - y         # Define function h
eq2 = sp.Eq(h,4)    # Define equation h = 4
eq2                 # print eq2
```
Out [22]: $2x - y = 4$

In [23]:
```
solns = sp.solve((eq1,eq2),(x,y))  # solve equations
solns                              # print solns
```
Out [23]: $\{x:\ 15/7,\ y:\ 2/7\}$

In [24]:
```
# The solution is given as a dictionary.
solns[x]    # print dictionary value for x
```
Out [24]: $\frac{15}{7}$

In [25]:
```
solns[y]    # print dictionary value for y
```
Out [25]: $\frac{2}{7}$

In [26]:
```
eq1.subs(((x,solns[x]),(y,solns[y])))   # check eq1
```
Out [26]: True

In [27]:
```
eq2.subs(((x,solns[x]),(y,solns[y])))   # check eq2
```
Out [27]: True

In [28]:
```
# Another example
# Define equations and solve
solns = solve((Eq(x**2 - y**2,0),Eq(x + 2*y,1)),(x,y))
solns        # print solutions
```
Out [28]: $[(-1,\ 1),\ (1/3,\ 1/3)]$

In [29]:
```
# The solution is given as a list of tuples
solns[0]     # print first solution
```
Out [29]: $(-1,\ 1)$

In [30]:
```
solns[1][0]  # print first element of second solution
```
Out [30]: $\frac{1}{3}$

Built–in functions:

In [31]:
```
myroot = sp.sqrt(x*y)    # square roots
myroot                   # print
```
Out [31]: $\sqrt{xy}$

In [32]:
```
mytrig = sp.sin(x)**2 - sp.cos(x)**2  # trig functions
sp.simplify(mytrig)                   # simplify and print
```
Out [32]: $-\cos(2x)$

In [33]:
```
myexp = sp.exp(x)*sp.exp(y)    # exponential functions
sp.simplify(myexp)             # simplify and print
```
Out [33]: e^{x+y}

In [34]:
```
mylog = sp.log(x) + sp.log(y)  # natural logarithms
mylog                          # print
```
Out [34]: $\log(x) + \log(y)$

Assumptions:

In [35]:
```
u = sp.symbols('u')                 # u is complex
v = sp.symbols('v', positive=True)  # v is positive
w = sp.symbols('w', real=True)      # w is real
```

In [36]:
```
sp.sqrt((u*v)**2)            # square root of (u*v)**2
```
Out [36]: $v\sqrt{u^2}$

In [37]:
```
eq1 = sp.Eq(u*(u**2 + 1),0)  # polynomial in u
sp.solve(eq1,u)              # solve eq1
```
Out [37]: $[0, -\mathtt{I}, \mathtt{I}]$

In [38]:
```
eq2 = sp.Eq(w*(w**2 + 1),0)  # same polynomial in w
sp.solve(eq2,w)              # solve eq2
```
Out [38]: $[0]$

Working with numbers:

In [39]:
```
sp.pi                  # pi
```
Out [39]: π

In [40]:
```
sp.pi.evalf()          # pi to 15 significant figures
```
Out [40]: 3.14159265358979

In [41]:
```
sp.pi.evalf(30)        # pi to 30 significant figures
```
Out [41]: 3.14159265358979323846264338328

In [42]:
```
1/3                           # regular Python calculation
```
Out [42]: 0.3333333333333333

In [43]:
```
sp.Integer(1)/sp.Integer(3)  # symbolic 1/3
```
Out [43]: $\frac{1}{3}$

In [44]:
```
sp.Rational(1,3)              # symbolic 1/3
```
Out [44]: $\frac{1}{3}$

In [45]:
```
# evaluate 1/3 to 12 significant figures
sp.Rational(1,3).evalf(12)
```
Out [45]: 0.333333333333

Exercise 10.1a

Define the polynomial

$$p = (x + y)(x^2 - y)(ax + y)$$

using SymPy.

- Expand p.
- Find the coefficient of x^3.
- Find the coefficient of a.
- Evaluate p at $x = 7$, $y = 3$.

Exercise 10.1b

Use SymPy to compute the following.

- Simplify the function $x - xz/y$.
- Factor the polynomial $x^3 - 2x^2 + x - 2$.
- Evaluate $\sin(e^x)$ at $x = \pi$ to 30 significant figures.

Exercise 10.1c

Solve the system of equations

$$2x^2 - 9y^2 = 1 \ ,$$
$$3y - x^2 = 0 \ ,$$

using SymPy. Check that the equations are satisfied by each solution.

10.2 Calculus with SymPy

In [1]:
```
import sympy as sp        # import sympy
x,y = sp.symbols('x y')   # define symbols

f = x*sp.cos(y) - y       # define a function f
f                         # print f
```
Out [1]: $x\cos(y) - y$

Derivatives:

In [2]:
```
sp.diff(f,y)    # differentiate f with respect to (wrt) y
```
Out [2]: $-x\sin(y) - 1$

In [3]:
```
sp.diff(f,x,y)  # differentiate f wrt x and y
```
Out [3]: $-\sin(y)$

In [4]:
```
myder = sp.Derivative(f,x) # symbolic derivative df/dx
myder                      # print
```
Out [4]: $\frac{\partial}{\partial x}(x\cos(y) - y)$

In [5]:
```
myder.doit()               # Evaluate symbolic derivative
```
Out [5]: $\cos(y)$

Integrals:

In [6]:
```
sp.integrate(f,x)        # integrate f wrt x
```
Out [6]: $\frac{x^2\cos(y)}{2} - xy$

In [7]:
```
sp.integrate(f,x,y)      # integrate f wrt x and y
```
Out [7]: $\frac{x^2 \sin(y)}{2} - \frac{xy^2}{2}$

In [8]:
```
sp.integrate(f,(y,-2,3)) # integrate f wrt y from -2 to 3
```
Out [8]: $x \sin(3) + x \sin(2) - \frac{5}{2}$

In [9]:
```
# integrate f wrt y from -2 to 3, wrt x from -1 to 1
sp.integrate(f,(y,-2,3),(x,-1,1))
```
Out [9]: -5

In [10]:
```
# symbolic integral of f wrt y
myint1 = sp.Integral(f,y)
myint1               # print
```
Out [10]: $\int (x \cos(y) - y)\, dy$

In [11]:
```
myint1.doit()        # evaluate symbolic integral
```
Out [11]: $x \sin(y) - \frac{y^2}{2}$

In [12]:
```
# symbolic integral of f wrt y from -1 to 1
myint2 = sp.Integral(f,(y,-1,1))
myint2               # print
```
Out [12]: $\int_{-1}^{1} (x \cos(y) - y)\, dy$

In [13]:
```
myint2.doit()        # evaluate symbolic integral
```
Out [13]: $2x \sin(1)$

Differential equations:

In [14]:
```
F = sp.Function('F')    # Define F as a symbolic function
t = sp.symbols('t')     # define t as a variable
```

In [15]:
```
sp.diff(F(t),t)         # derivative of F(t) wrt t
```
Out [15]: $\frac{d}{dt} F(t)$

In [16]:
```
# create a differential equation
myde = sp.Eq(sp.diff(F(t),t,t) - F(t), sp.exp(t))
myde                      # print
```

Out [16]: $-F(t) + \frac{d^2}{dt^2}F(t) = e^t$

In [17]:
```
sp.dsolve(myde,F(t))    # solve the differential equation
```

Out [17]: $F(t) = C_2 e^{-t} + \left(C_1 + \frac{t}{2}\right) e^t$

Limits and series:

In [18]:
```
f = sp.sin(2*x)/x      # define f
f                      # print f
```

Out [18]: $\frac{\sin(2x)}{x}$

In [19]:
```
sp.limit(f,x,0)        # limit of f(x) as x goes to 0
```

Out [19]: 2

In [20]:
```
# Taylor series for f(x) about x=0 through order x**7
myseries = sp.series(f,x,0,8)
myseries              # print
```

Out [20]: $2 - \frac{4x^2}{3} + \frac{4x^4}{15} - \frac{8x^6}{315} + \mathcal{O}(x^8)$

In [21]:
```
myseries.removeO()    # remove the O(x**8) symbol
```

Out [21]: $\frac{8x^6}{315} + \frac{4x^4}{15} - \frac{4x^2}{3} + 2$

Exercise 10.2a

Consider the function

$$f(x, y) = \sin(x + y)\cos(2x - y) \ .$$

Using SymPy,

- Differentiate $f(x, y)$ with respect to x and simplify the result.
- Integrate $f(x, y)$ with respect to x and simplify the result.

- Integrate $f(x, y)$ with respect to x from $x = -1$ to $x = 1$, substitute the value $y = 2$, and evaluate the answer numerically to 8 significant figures.

Exercise 10.2b

Solve the differential equation

$$x^2 \frac{d^2}{dx^2} F(x) - x \frac{d}{dx} F(x) - 3F(x) = 4x^3$$

using SymPy.

Exercise 10.2c

With SymPy, compute the Taylor series for

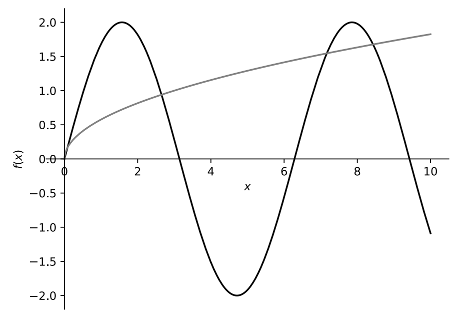
$$f(x) = \ln(x + \cos(x)) \ ,$$

about $x = 0$. Include terms up to x^7. Remove the $\mathcal{O}(x^8)$ symbol and evaluate the expression at $x = 1/2$. Compare the series approximation to $f(1/2)$.

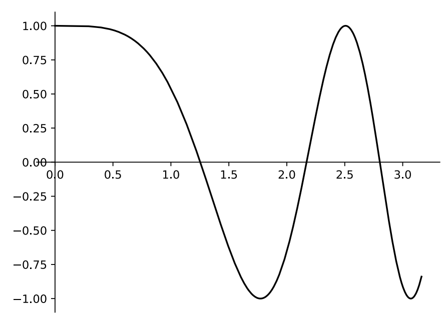
10.3 Plotting graphs with SymPy

In [1]:
```
import sympy as sp            # import sympy
x,y,u = sp.symbols('x y u')   # define symbols
```

In [2]:
```
# plot 2*sin(x) and sqrt(x/3) for x from 0 to 10
sp.plot(2*sin(x), sqrt(x/3), (x,0,10))
```



In [3]:
```
# parametric plot of x = sqrt(u), y = cos(u)
# for u from 0 to 10
sp.plot_parametric((sqrt(u),cos(u)), (u,0,10))
```



Exercise 10.3a

Use SymPy to plot (on the same graph) the functions $\sin(x)$ and $\sin((\pi/2)\sin(x))$ over the domain $-2\pi \le x \le 2\pi$.

Exercise 10.3b

A cycloid is the path made by a point on a circle as the circle rolls along the x–axis. For a circle of radius R, a cycloid is defined parametrically by the equations

$$x = R(t - \sin t) \ ,$$
$$y = R(1 - \cos t) \ .$$

Use SymPy to plot the cycloid with $R = 1$.

10.4 Linear algebra with SymPy

In [1]:
```
import sympy as sp            # import sympy
x,y = sp.symbols('x y')       # define symbols
```

Matrices:

In [2]:
```
A = sp.Matrix([[1,3],[2,5]])  # Define a 2x2 matrix A
A                             # print A
```

Out [2]: $\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix}$

In [3]:
```
A[1,0]        # second row, first column
```

Out [3]: 2

In [4]:
```
A[0,1]        # first row, second column
```

Out [4]: 3

In [5]:
```
det(A)        # determinant of A
```

Out [5]: -1

In [6]:
```
A.inv()       # inverse of A
```

Out [6]: $\begin{bmatrix} -5 & 3 \\ 2 & -1 \end{bmatrix}$

In [7]:
```
sp.eye(3)     # 3x3 identity matrix
```

Out [7]: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

In [8]:
```
B = sp.Matrix([[3*x,1],[2,-y]])  # another matrix
B                                # print
```

Out [8]: $\begin{bmatrix} 3x & 1 \\ 2 & -y \end{bmatrix}$

In [9]:
```
A*B               # multiply matrices
```

Out [9]: $\begin{bmatrix} 3x+6 & 1-3y \\ 6x+10 & 2-5y \end{bmatrix}$

In [10]:
```
A - B             # subtract (or add) matrices
```

Out [10]: $\begin{bmatrix} 1-3x & 2 \\ 0 & y+5 \end{bmatrix}$

Matrix equations:

In [11]:
```
b =  sp.Matrix([[2],[3]])    # column vector
b                            # print b
```

Out [11]: $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$

In [12]:
```
v = A.solve(b) # solve the matrix equation A*v = b for v
v              # print v
```

Out [12]: $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$

Vectors:

In [13]:
```
v1 = sp.Matrix([1,2,3])   # define a vector
v2 = sp.Matrix([4,5,6])   # another vector
v2                        # print v2
```

Out [13]: $\begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$

In [14]:
```
v1.dot(v2)        # dot product v1.v2
```

Out [14]: 32

In [15]:
```
v1.cross(v2)     # cross product v1 x v2
```

Out [15]:

$$\begin{bmatrix} -3 \\ 6 \\ -3 \end{bmatrix}$$

Exercise 10.4a

Use SymPy to define the matrices

$$M = \begin{pmatrix} 3 & -5 & 2 \\ -1 & 4 & 4 \\ 2 & 3 & -2 \end{pmatrix} ,$$

$$N = \begin{pmatrix} 1 & 3 & -2 \\ 2 & 5 & -3 \\ 4 & -3 & -2 \end{pmatrix} .$$

Compute the product MN. Find the determinant of MN and the inverse of MN.

Exercise 10.4b

Let

$$A = \begin{pmatrix} 2 & 2 & -4 \\ -1 & 2 & -3 \\ -2 & 1 & 1 \end{pmatrix} ,$$

$$b = \begin{pmatrix} 3 \\ 5 \\ 1 \end{pmatrix} .$$

With SymPy, solve the matrix equation $Av = b$ for the vector v.

Exercise 10.4c

Use SymPy to define the vector

$$v_1 = \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix} ,$$

along with two more vectors v_2 and v_3. Verify the triple product identity $v_1 \times (v_2 \times v_3) = (v_1 \cdot v_3)v_2 - (v_1 \cdot v_2)v_3$.

10.5 From SymPy to NumPy

In [1]:
```
import sympy as sp            # import sympy
import numpy as np            # import numpy
x = sp.symbols('x')           # define variables
```

In [2]:
```
f = sp.sin(sp.log(x))         # define a sympy function
f                             # print
```

Out [2]: $\sin(\log(x))$

In [3]:
```
f = sp.lambdify(x,f,"numpy") # convert to numpy function
f(2)                          # evaluate f at 2
```

Out [3]: 0.6389612763136348

In [4]:
```
x = np.linspace(2,5,4)        # create array of x values
f(x)                          # evaluate f at x
```

Out [4]: array([0.63896128, 0.89057704, 0.98302774, 0.99925351])

Exercise 10.5

Use SymPy to compute

$$f(x) = \int x \sin^2(x)\, dx \ .$$

Convert $f(x)$ to a NumPy function, then use Matplotlib to plot the function.

10.6 Printing with SymPy

A final word on printing with SymPy. When we type a variable name such as `f` (or a command such as `factor(f)`) into the last line of a jupyter notebook cell, Python responds by printing the result to the screen. You can always use the `print()` function to tell Python to print. However, the output might be less readable when you use `print()` explicitly. For example, consider the function $f = x^2 - y^2$. The output from `f` is

$$x^2 - y^2 \ ,$$

whereas the output from `print(f)` is

$$\texttt{x**2} - \texttt{y**2} \ .$$

SymPy also has a “pretty print” feature. The command `pprint(f)` produces

$$\mathtt{x}^2 - \mathtt{y}^2 .$$

These results may differ on different platforms.