S2

June 11, 2021

1 Symbolic Computing

[1]:
```
import sympy
from sympy import I, pi, oo
```

1.1 Symbols

[2]:
```
# sympy.Symbol, sympy.symbols, sympy.var
# real, imaginary, positive, negative, integer, odd, even, prime, finite,␣
↪infinite
```

[3]:
```
x = sympy.Symbol("x")
x
```

[3]:

x

[4]:
```
y = sympy.Symbol("y", real=True)
y.is_real
```

[4]: True

[5]:
```
x.is_real is None
```

[5]: True

[6]:
```
x = sympy.Symbol("x")
y = sympy.Symbol("y", positive=True)
```

[7]:
```
sympy.sqrt(x**2)
```

[7]:

$\sqrt{x^2}$

[8]:
```
sympy.sqrt(y**2)
```

[8]:

y

to get better answers from Sympy, we should determine every detail.

[9]:
```python
n1 = sympy.Symbol("n")
n2 = sympy.Symbol("n", integer=True)
n3 = sympy.Symbol("n", odd=True)
```

[10]:
```python
sympy.cos(n1 * pi)
```

[10]: $\cos(\pi n)$

[11]:
```python
sympy.cos(n2 * pi)
```

[11]: $(-1)^n$

[12]:
```python
sympy.cos(n3 * pi)
```

[12]: -1

1.1.1 Numbers

[13]:
```python
#sympy.Symbol("a", integer=True), sympy.Integer(), is_Name, is_name
```

an integer with unknown value.

[14]:
```python
n = sympy.Symbol("n", integer=True)
```

[15]:
```python
n.is_integer, n.is_Integer, n.is_positive, n.is_Symbol
```

[15]:
```
(True, False, None, True)
```

an integer with specific value.

[16]:
```python
i = sympy.Integer(19)
```

[17]:
```python
i.is_integer, i.is_Integer, i.is_positive, i.is_Symbol
```

[17]:
```
(True, True, True, False)
```

[18]:
```python
"%0.25f" % 0.3
```

[18]:
```
'0.2999999999999999888977698'
```

[19]:
```python
sympy.Float(0.3, 25)
```

[19]: 0.2999999999999999888977698

[20]:
```
sympy.Float('0.3', 25)     # exact value
```

[20]: 0.3

[21]:
```
sympy.Rational(21, 55)
```

[21]: $\frac{21}{55}$

[22]:
```
r1 = sympy.Rational(2, 5)
r2 = sympy.Rational(4, 9)
```

[23]:
```
r1 * r2
```

[23]: $\frac{8}{45}$

[24]:
```
r1 / r2
```

[24]: $\frac{9}{10}$

[25]:
```
x, y, z = sympy.symbols("x, y, z")
```

undefined function

[26]:
```
f = sympy.Function("f")
f(x)
```

[26]: $f(x)$

[27]:
```
g = sympy.Function("g")(x, y, z)
g
```

[27]: $g(x, y, z)$

[28]:
```
g.free_symbols
```

[28]:
```
{x, y, z}
```

[29]:
```
sympy.sin
```

[29]:
```
sin
```

[30]:
```
sympy.sin(x)
```

[30]:

$\sin(x)$

[31]:
```
sympy.sin(pi * 1.5)
```

[31]:

-1

[32]:
```
h = sympy.Lambda(x, x**3)
h
```

[32]:

$(x \mapsto x^3)$

[33]:
```
h(2)
```

[33]:

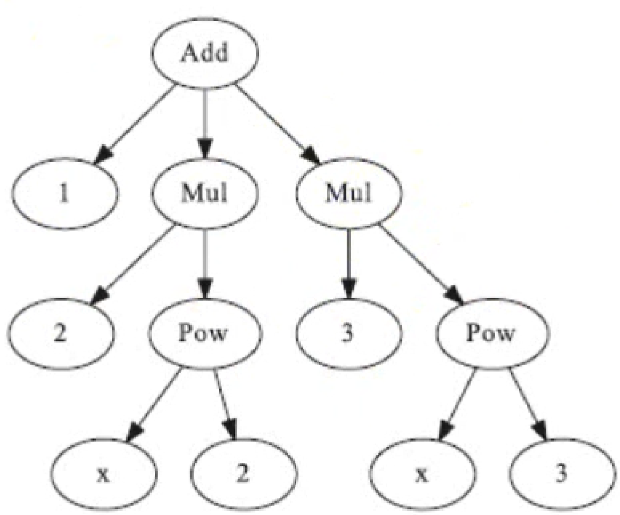
8

[34]:
```
h(1+x)
```

[34]:

$(x+1)^3$

1.2 Expressions



[35]:
```
x = sympy.Symbol("x")
```

[36]:
```
expr = 1 + 2 * x ** 2 + 3 * x ** 3
expr
```

[36]:

$3x^3 + 2x^2 + 1$

[37]:
```
expr.args
```

[37]: (1, 2*x**2, 3*x**3)

[38]:
```
expr.args[2]
```

[38]: $3x^3$

[39]:
```
expr.args[2].args[1]
```

[39]: x^3

[40]:
```
expr.args[2].args[1].args[0]
```

[40]: x

[41]:
```
expr.args[2].args[1].args[0].args
```

[41]: ()

1.3 Manipulating Expressions

1.3.1 Simplification

[42]:
```
expr = 2 * (x**2 - x) - x * (x + 1)
expr
```

[42]: $2x^2 - x(x+1) - 2x$

[43]:
```
sympy.simplify(expr)
```

[43]: $x(x-3)$

expr doesn't change.

[44]:
```
expr
```

[44]: $2x^2 - x(x+1) - 2x$

[45]:
```
expr.simplify()
```

[45]: $x(x-3)$

[46]:
```
expr = 2 * sympy.cos(x) * sympy.sin(x)
expr
```

[46]: $2\sin(x)\cos(x)$

[47]:
```
sympy.simplify(expr)
```

[47]: $\sin(2x)$

[48]:
```python
expr = sympy.exp(x) * sympy.exp(y)
expr
```

[48]:

$e^{x}e^{y}$

[49]:
```python
sympy.simplify(expr)
```

[49]:

e^{x+y}

some special symplifications.

[50]:
```python
# sympy.trigsimp, sympy.powsimp, sympy.compsimp
```

1.3.2 Expand

[51]:
```python
expr = (x + 1) * (x + 2)
```

[52]:
```python
sympy.expand(expr)
```

[52]:

$x^2 + 3x + 2$

[53]:
```python
sympy.sin(x + y).expand()
```

[53]:

$\sin(x + y)$

[54]:
```python
sympy.sin(x + y).expand(trig=True)
```

[54]:

$\sin(x)\cos(y) + \sin(y)\cos(x)$

[55]:
```python
a, b = sympy.symbols("a, b", positive=True)
```

[56]:
```python
sympy.log(a * b).expand(log=True)
```

[56]:

$\log(a) + \log(b)$

[57]:
```python
sympy.exp(I*a + b).expand(complex=True)
```

[57]:

$ie^{b}\sin(a) + e^{b}\cos(a)$

[58]:
```python
sympy.expand((a * b)**x, power_base=True)
```

[58]:

$a^{x}b^{x}$

[59]:
```python
sympy.exp((a-b)*x).expand(power_exp=True)
```

[59]:

$e^{ax}e^{-bx}$

special expand methods.

[60]:
```
# sympy.expand_mul, sympy.expand_trig, sympy.expand_log,
# sympy.expand_complex, sympy.expand_power_base, sympy.expand_power_exp
```

1.3.3 Factor, Collect, and Combine

[61]:
```
sympy.factor(x**2 - 1)
```

[61]: $(x - 1)(x + 1)$

[62]:
```
sympy.factor(x * sympy.cos(y) + sympy.sin(z) * x)
```

[62]: $x(\sin(z) + \cos(y))$

[63]:
```
sympy.logcombine(sympy.log(a) - sympy.log(b))
```

[63]: $\log\left(\frac{a}{b}\right)$

[64]:
```
expr = x + y + x * y * z
```

[65]:
```
expr.collect(x)
```

[65]: $x(yz + 1) + y$

[66]:
```
expr.collect(y)
```

[66]: $x + y(xz + 1)$

[67]:
```
sympy.collect(expr, x)
```

[67]: $x(yz + 1) + y$

[68]:
```
expr.collect([y, x])
```

[68]: $x + y(xz + 1)$

[69]:
```
expr.collect([x, y])
```

[69]: $x(yz + 1) + y$

[70]:
```
expr = sympy.cos(x + y) + sympy.sin(x - y)
expr
```

[70]: $\sin(x - y) + \cos(x + y)$

[71]:
```
expr.expand(trig=True).collect([sympy.cos(x), sympy.sin(x)]).collect(sympy.
 ↪cos(y) - sympy.sin(y))
```

[71]: $(\sin(x) + \cos(x))(-\sin(y) + \cos(y))$

1.3.4 Apart, Together(inverse of apart), and Cancel

used for manipulating fractions.

[72]:
```
sympy.apart(1/(x**2 + 3*x + 2), x)
```

[72]:
$$-\frac{1}{x+2}+\frac{1}{x+1}$$

[73]:
```
sympy.apart(1/(x**2 + 3*x + 2))
```

[73]:
$$-\frac{1}{x+2}+\frac{1}{x+1}$$

[74]:
```
sympy.together(1 / (y * x + y) + 1 / (1+x))
```

[74]:
$$\frac{y+1}{y\,(x+1)}$$

[75]:
```
sympy.cancel(y / (y * x + y))
```

[75]:
$$\frac{1}{x+1}$$

1.3.5 Substitutions

[76]:
```
# subs(popular), replace
```

[77]:
```
(x + y).subs(x, y)
```

[77]:
$2y$

[78]:
```
sympy.sin(x * z + 3*y).subs({x : y**2, z : sympy.exp(y), y : y**4, sympy.sin :␣
↪sympy.cos})
```

[78]:
$\cos\left(y^{8}e^{y}+3y^{4}\right)$

[79]:
```
expr = x * z + y**3 + y * x**2
expr.subs({x : 1.2, y : 0.78, z : 3.5})
```

[79]:
5.797752

1.4 Numerical Evaluation

[80]:
```
sympy.N(1 + pi)
```

[80]:
4.14159265358979

[81]:
```python
sympy.N(pi, 50)
```

[81]:

$$3.1415926535897932384626433832795028841971693993751$$

[82]:
```python
(x + 1/pi).evalf(10)
```

[82]:

$$x + 0.3183098862$$

[83]:
```python
expr = sympy.sin(pi * x * sympy.exp(x))
```

[84]:
```python
[expr.subs(x, xx).evalf(4) for xx in range(0, 10)]
```

[84]:
```
[0, 0.7739, 0.6420, 0.7216, 0.9436, 0.2052, 0.9740, 0.9773, -0.8703, -0.6951]
```

isntead of using `for`, we use lambdify.
lambdify is more efficient and returns a function.

[85]:
```python
expr_func = sympy.lambdify(x, expr)
expr_func(2.0)
```

[85]:
```
0.6419824398474936
```

we can use array as parameter for lambdify.

[86]:
```python
import numpy as np
```

[87]:
```python
xval = np.arange(0, 10)
expr_func(xval)
```

[87]:
```
array([ 0.        ,  0.77394269,  0.64198244,  0.72163867,  0.94361635,
        0.20523391,  0.97398794,  0.97734066, -0.87034418, -0.69512687])
```

1.5 Calculus

1.5.1 Derivatives

[88]:
```python
f = sympy.Function('f')(x)
f
```

[88]:

$$f(x)$$

[89]:
```python
sympy.diff(f, x)
```

[89]:

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

[90]:
```python
f.diff(x)
```

[90]:

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

[91]:
```
sympy.diff(f, x, x)
```

[91]:

$$\frac{d^2}{dx^2} f(x)$$

[92]:
```
sympy.diff(f, x, 2)
```

[92]:

$$\frac{d^2}{dx^2} f(x)$$

[93]:
```
sympy.diff(f, x, 3)
```

[93]:

$$\frac{d^3}{dx^3} f(x)$$

[94]:
```
g = sympy.Function('g')(x, y)
```

[95]:
```
g.diff(x)
```

[95]:

$$\frac{\partial}{\partial x} g(x, y)$$

[96]:
```
g.diff(x, y)
```

[96]:

$$\frac{\partial^2}{\partial y \partial x} g(x, y)$$

[97]:
```
g.diff(x, 2, y, 3)
```

[97]:

$$\frac{\partial^5}{\partial y^3 \partial x^2} g(x, y)$$

[98]:
```
expr = x**4 + 3*x**3 + 4*x**2 + 8
```

[99]:
```
expr.diff(x)
```

[99]:

$$4x^3 + 9x^2 + 8x$$

[100]:
```
expr.diff(x, 3)
```

[100]:

$$6(4x + 3)$$

[101]:
```
expr = (x+1)**3 * y**2 * (z-8)
```

[102]:
```
expr.diff(x, y, z)
```

[102]:

$$6y(x + 1)^2$$

[103]:
```
expr = sympy.sin(x * y) * sympy.cos(x / 5)
expr
```

[103]:

$$\sin(xy)\cos\left(\frac{x}{5}\right)$$

[104]:
```python
expr.diff(x)
```

[104]:

$$y\cos\left(\frac{x}{5}\right)\cos(xy) - \frac{\sin\left(\frac{x}{5}\right)\sin(xy)}{5}$$

for showing $\frac{d}{dx}$, use `Derivative`.

[105]:
```python
d = sympy.Derivative(sympy.exp(sympy.cos(x)), x)
d
```

[105]:

$$\frac{d}{dx}e^{\cos(x)}$$

[106]:
```python
d.doit()
```

[106]:

$$-e^{\cos(x)}\sin(x)$$

1.5.2 Integrals

many integrals, can't be solved by analytic methods.

[107]:
```python
# sympy.integrate, sympy.Integral
```

[108]:
```python
a, b, x, y = sympy.symbols("a, b, x, y")
```

[109]:
```python
f = sympy.Function("f")(x)
f
```

[109]:

$$f(x)$$

[110]:
```python
sympy.integrate(f)
```

[110]:

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

[111]:
```python
sympy.integrate(f, (x, a, b))
```

[111]:

$$\int\limits_a^b f(x)\,dx$$

[112]:
```python
g = sympy.Function("g")(x, y)
```

[113]:
```python
sympy.integrate(g, (x, a, b), (y, a, b))
```

[113]:

$$\int\limits_a^b\int\limits_a^b g(x,y)\,dx\,dy$$

[114]:
```python
sympy.integrate(sympy.sin(x))
```

[114]:

$$-\cos(x)$$

[115]:
```
sympy.integrate(sympy.sin(x), (x, a, b))
```

[115]:

$$\cos(a) - \cos(b)$$

[116]:
```
sympy.integrate(sympy.exp(-x**2), (x, 0, oo))
```

[116]:

$$\frac{\sqrt{\pi}}{2}$$

[117]:
```
a, b, c = sympy.symbols("a, b, c", positive=True)
```

[118]:
```
sympy.integrate(a * sympy.exp(-((x-b)/c)**2), (x, -oo, oo))
```

[118]:

$$\sqrt{\pi}ac$$

can't be solved.

[119]:
```
sympy.integrate(sympy.sin(x * sympy.cos(x)))
```

[119]:

$$\int \sin(x\cos(x))\, dx$$

multi variable expressions.

[120]:
```
expr = sympy.sin(x*sympy.exp(y))
expr
```

[120]:

$$\sin(xe^{y})$$

[121]:
```
sympy.integrate(expr, x)
```

[121]:

$$-e^{-y}\cos(xe^{y})$$

[122]:
```
sympy.integrate(expr, x,  y)
```

[122]:

$$x\operatorname{Si}(xe^{y}) + e^{-y}\cos(xe^{y})$$

[123]:
```
sympy.integrate(expr, (x, 0, 1), (y, 0, 1))
```

[123]:

$$-\operatorname{Si}(1) - \cos(1) - e^{-1} + \frac{\cos(e)}{e} + 1 + \operatorname{Si}(e)$$

[124]:
```
s = sympy.Integral(expr, x)
s
```

[124]:

$$\int \sin(xe^{y})\, dx$$

[125]:
```
s.doit()
```

[125]:

$$-e^{-y}\cos(xe^{y})$$

[126]:
```
ss = sympy.Integral(expr, x, y)
```

[127]:
```
ss
```

[127]:

$$\iint \sin\left(xe^{y}\right)\,dx\,dy$$

[128]:
```
ss.doit()
```

[128]:

$$x\operatorname{Si}\left(xe^{y}\right) + e^{-y}\cos\left(xe^{y}\right)$$

1.5.3 Series

[129]:
```
# defaults: x0=0, n=6, dir='+'
```

[130]:
```
x, y = sympy.symbols("x, y")
f = sympy.Function("f")(x)
```

[131]:
```
sympy.series(f, x)
```

[131]:

$$f(0) + x\left.\frac{d}{dx}f(x)\right|_{x=0} + \frac{x^{2}\left.\frac{d^{2}}{dx^{2}}f(x)\right|_{x=0}}{2} + \frac{x^{3}\left.\frac{d^{3}}{dx^{3}}f(x)\right|_{x=0}}{6} + \frac{x^{4}\left.\frac{d^{4}}{dx^{4}}f(x)\right|_{x=0}}{24} + \frac{x^{5}\left.\frac{d^{5}}{dx^{5}}f(x)\right|_{x=0}}{120} + O\left(x^{6}\right)$$

[132]:
```
x0 = sympy.Symbol("{x_0}")
```

[133]:
```
f.series(x, x0, n=4)
```

[133]:

$$f(x_0) + (x - x_0)\left.\frac{d}{d\xi_1}f(\xi_1)\right|_{\xi_1=x_0} + \frac{(x-x_0)^{2}\left.\frac{d^{2}}{d\xi_1^{2}}f(\xi_1)\right|_{\xi_1=x_0}}{2} + \frac{(x-x_0)^{3}\left.\frac{d^{3}}{d\xi_1^{3}}f(\xi_1)\right|_{\xi_1=x_0}}{6} + O\left((x-x_0)^{4}; x \to x_0\right)$$

[134]:
```
f.series(x, x0, n=4).removeO()
```

[134]:

$$\frac{(x-x_0)^{3}\left.\frac{d^{3}}{d\xi_1^{3}}f(\xi_1)\right|_{\xi_1=x_0}}{6} + \frac{(x-x_0)^{2}\left.\frac{d^{2}}{d\xi_1^{2}}f(\xi_1)\right|_{\xi_1=x_0}}{2} + (x - x_0)\left.\frac{d}{d\xi_1}f(\xi_1)\right|_{\xi_1=x_0} + f(x_0)$$

[135]:
```
sympy.cos(x).series()
```

[135]:

$$1 - \frac{x^{2}}{2} + \frac{x^{4}}{24} + O\left(x^{6}\right)$$

[136]:
```
sympy.sin(x).series()
```

[136]:

$$x - \frac{x^{3}}{6} + \frac{x^{5}}{120} + O\left(x^{6}\right)$$

[137]:
```
sympy.exp(x).series()
```

[137]:

$$1 + x + \frac{x^{2}}{2} + \frac{x^{3}}{6} + \frac{x^{4}}{24} + \frac{x^{5}}{120} + O\left(x^{6}\right)$$

[138]:
```
(1/(1+x)).series()
```

[138]:
$$1 - x + x^2 - x^3 + x^4 - x^5 + O\left(x^6\right)$$

[139]:
```
expr = sympy.sin(x) / (1 + sympy.cos(x * y))
expr
```

[139]:
$$\frac{\sin\left(x\right)}{\cos\left(xy\right) + 1}$$

[140]:
```
expr.series(x, n=4)
```

[140]:
$$\frac{x}{2} + x^3\left(\frac{y^2}{8} - \frac{1}{12}\right) + O\left(x^4\right)$$

[141]:
```
expr.series(y, n=4)
```

[141]:
$$\frac{\sin\left(x\right)}{2} + \frac{x^2y^2\sin\left(x\right)}{8} + O\left(y^4\right)$$

1.5.4 Limits

[142]:
```
sympy.limit(sympy.sin(x)/x, x, 0)
```

[142]:
$$1$$

derivative

[143]:
```
f = sympy.Function('f')
x, h = sympy.symbols("x, h")
diff_limit = (f(x+h) - f(x))/h
```

[144]:
```
sympy.limit(diff_limit.subs(f, sympy.sin), h, 0)
```

[144]:
$$\cos\left(x\right)$$

Asymptotic behavior

[145]:
```
expr = (x**2 - 4*x) / (3*x -5)
expr
```

[145]:
$$\frac{x^2 - 4x}{3x - 5}$$

[146]:
```
# f(x) -> px + q
```

[147]:
```
p = sympy.limit(expr/x, x, oo)
q = sympy.limit(expr - p*x, x, oo)
p, q
```

[147]: (1/3, -7/9)

[148]:
```
# f(x) -> x/3 - 7/9
```

1.5.5 Sums and Products

[149]:
```
n = sympy.Symbol("n", integer=True)
x = sympy.Sum(1/(n**2), (n, 1, oo))
x
```

[149]: $$\sum_{n=1}^{\infty} \frac{1}{n^2}$$

[150]:
```
x.doit()
```

[150]: $$\frac{\pi^2}{6}$$

[151]:
```
x = sympy.Product(n, (n, 1, 10))
x
```

[151]: $$\prod_{n=1}^{10} n$$

[152]:
```
x.doit()
```

[152]: 3628800

[153]:
```
x = sympy.Symbol("x")
sympy.Sum((x)**n/(sympy.factorial(n)), (n, 1, oo))
```

[153]: $$\sum_{n=1}^{\infty} \frac{x^n}{n!}$$

[154]:
```
sympy.Sum((x)**n/(sympy.factorial(n)), (n, 1, oo)).doit().simplify()
```

[154]: $e^x - 1$

1.6 Equations

many equations can't be solved analytically.

[155]:
```
x = sympy.Symbol("x")
expr = x**2 + 2*x - 3
```

[156]:
```
sympy.solve(expr)
```

[156]: [-3, 1]

[157]:
```
a, b, c = sympy.symbols("a, b, c")
expr = a*x**2 + b*x + c
expr
```

[157]: $ax^2 + bx + c$

[158]:
```
sympy.solve(expr, x)
```

[158]: [(-b + sqrt(-4*a*c + b**2))/(2*a), -(b + sqrt(-4*a*c + b**2))/(2*a)]

[159]:
```
sympy.solve(sympy.sin(x) - sympy.cos(x), x)
```

[159]: [pi/4]

can't solve.

[160]:
```
sympy.solve(x**5 - x**2 + 1, x)
```

[160]:
```
[CRootOf(x**5 - x**2 + 1, 0),
 CRootOf(x**5 - x**2 + 1, 1),
 CRootOf(x**5 - x**2 + 1, 2),
 CRootOf(x**5 - x**2 + 1, 3),
 CRootOf(x**5 - x**2 + 1, 4)]
```

[161]:
```
# sympy.solve(sympy.tan(x) + x, x)    # can't solve and returns ERROR.
```

Equations System.

[162]:
```
eq1 = x + 2*y - 4
eq2 = x - y + 2
```

[163]:
```
sympy.solve([eq1, eq2], [x, y], dict=True)
```

[163]: [{x: 0, y: 2}]

[164]:
```
eq1 = x**2 - y
eq2 = y**2 - x
```

[165]:
```
sols = sympy.solve([eq1, eq2], [x, y], dict=True)
sols
```

[165]:
```
[{x: 0, y: 0},
 {x: 1, y: 1},
 {x: (-1/2 - sqrt(3)*I/2)**2, y: -1/2 - sqrt(3)*I/2},
 {x: (-1/2 + sqrt(3)*I/2)**2, y: -1/2 + sqrt(3)*I/2}]
```

checking the roots

[166]:
```
[eq1.subs(sol).simplify() == 0 and eq2.subs(sol).simplify() == 0 for sol in␣
↪sols]
```

[166]: [True, True, True, True]

1.7 Linear Algebra

supports up to 2D Matrices.

[167]:
```
sympy.Matrix([1, 2])
```

[167]:
$$\begin{bmatrix}1\\2\end{bmatrix}$$

[168]:
```
sympy.Matrix([[1, 2]])
```

[168]:
$$\begin{bmatrix}1 & 2\end{bmatrix}$$

[169]:
```
sympy.Matrix([[1, 2], [3, 4]])
```

[169]:
$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$$

[170]:
```
sympy.Matrix(3, 4, lambda m, n: 10*m + n)
```

[170]:
$$\begin{bmatrix}0 & 1 & 2 & 3\\10 & 11 & 12 & 13\\20 & 21 & 22 & 23\end{bmatrix}$$

for numeric caculations, use `NumPy`.
for parametric caculations, use `SymPy`.

[171]:
```
a, b, c, d = sympy.symbols("a, b, c, d")
M = sympy.Matrix([[a, b], [c, d]])
M
```

[171]:
$$\begin{bmatrix}a & b\\c & d\end{bmatrix}$$

[172]:
```
M * M
```

[172]:
$$\begin{bmatrix}a^2+bc & ab+bd\\ac+cd & bc+d^2\end{bmatrix}$$

[173]:
```
x = sympy.Matrix(sympy.symbols("x_1, x_2"))
x
```

[173]:
$$\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

[174]:
```
M * x
```

[174]:

$$\begin{bmatrix} ax_1 + bx_2 \\ cx_1 + dx_2 \end{bmatrix}$$

[175]:
```
# sympy.transpose(M), M.transpose(), M.T(),
# trace, det, inv
```

[176]:
```
M
```

[176]:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

[177]:
```
M.T
```

[177]:

$$\begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

[178]:
```
sympy.transpose(M)
```

[178]:

$$\begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

[179]:
```
M.inv()
```

[179]:

$$\begin{bmatrix} \frac{d}{ad-bc} & -\frac{b}{ad-bc} \\ -\frac{c}{ad-bc} & \frac{a}{ad-bc} \end{bmatrix}$$

[180]:
```
M.det()
```

[180]:

$$ad - bc$$

[181]:
```
M.trace()
```

[181]:

$$a + d$$

[182]:
```
M.transpose()
```

[182]:

$$\begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

[183]:
```
p, q = sympy.symbols("p, q")
```

[184]:
```
M = sympy.Matrix([[1, p], [q, 1]])
```

[185]:
```
M
```

[185]:

$$\begin{bmatrix} 1 & p \\ q & 1 \end{bmatrix}$$

[186]:
```
b = sympy.Matrix(sympy.symbols("b_1, b_2"))
```

[187]:
```
b
```

[187]:

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

[188]:
```
# Mx = b
```

[189]:
```
x = M.LUsolve(b)
```

[190]:
```
x
```

[190]:

$$\begin{bmatrix} b_1 - \frac{p(-b_1q+b_2)}{-pq+1} \\ \frac{-b_1q+b_2}{-pq+1} \end{bmatrix}$$

[191]:
```
x = M.inv() * b
```

[192]:
```
x
```

[192]:

$$\begin{bmatrix} \frac{b_1}{-pq+1} - \frac{b_2p}{-pq+1} \\ -\frac{b_1q}{-pq+1} + \frac{b_2}{-pq+1} \end{bmatrix}$$