ch_13_assignment

April 11, 2023

Ch_13_assignment

[ ]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
```

1 Symbolic Computing

1.1 Importing Sympy

[ ]:
```
import sympy
sympy.init_printing()
from sympy import I, pi, oo
```

1.2 Symbols

[ ]:
```
x = sympy.Symbol("x")
y = sympy.Symbol("y", real=True)
z = sympy.Symbol("z", imaginary=True)

x.is_real is None
y.is_real
z.is_real
```

[ ]: True

[ ]: True

[ ]: False

[ ]:
```
x = sympy.Symbol("x")
y = sympy.Symbol("y", positive=True)
sympy.sqrt(x ** 2)
sympy.sqrt(y ** 2)
```

[ ]: $\sqrt{x^2}$

[ ]:

y

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

sympy.cos(n1 * pi)
sympy.cos(n2 * pi)
sympy.cos(n3 * pi)
```

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

[ ]: $(-1)^n$

[ ]: -1

[ ]:
```python
a, b, c = sympy.symbols("a, b, c", negative=True)
d, e, f = sympy.symbols("d, e, f", positive=True)
```

1.3 Numbers

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

i.is_integer, i.is_real, i.is_odd

f = sympy.Float(2.3)
type(f)

f.is_Integer, f.is_real, f.is_odd
```

[ ]: `sympy.core.numbers.Integer`

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

[ ]: `sympy.core.numbers.Float`

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

[ ]:
```python
i, f = sympy.sympify(19), sympy.sympify(2.3)
type(i), type(f)
```

[ ]: `(sympy.core.numbers.Integer, sympy.core.numbers.Float)`

1.3.1 Integer

[ ]:
```python
i ** 50
```

[ ]: $8663234049605954426644038200675212212900743262211018069459689001$

1.3.2 Float

[ ]:
```python
"%.25f" % 0.3
sympy.Float(0.3, 25)
sympy.Float('0.3', 25)
```

[ ]: '0.2999999999999999888977698'

[ ]: $0.2999999999999999888977698$

[ ]: 0.3

1.3.3 Rational

[ ]:
```python
sympy.Rational(11, 13)
r1 = sympy.Rational(2, 3)
r2 = sympy.Rational(4, 5)
r1 * r2
```

[ ]: $\frac{11}{13}$

[ ]: $\frac{8}{15}$

1.3.4 Functions

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

f(x)
```

[ ]: sympy.core.function.UndefinedFunction

[ ]: $f(x)$

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

g.free_symbols
```

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

[ ]: $\{x, y, z\}$

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

n = sympy.Symbol("n", integer=True)
```

```
sympy.sin(pi * n)
```

[ ]: sin

[ ]: $\sin(x)$

[ ]: -1

[ ]: 0

[ ]:
```
h = sympy.Lambda(x, x**2)
h

h(5)
h(1+x)
```

[ ]: $(x \mapsto x^2)$

[ ]: 25

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

1.4 Experssions

[ ]:
```
x = sympy.Symbol("x")
expr = 1 + 2 * x**2 + 3 * x ** 3
expr
```

[ ]: $3x^3 + 2x^2 + 1$

1.5 Manipulating Expression

1.5.1 Simplification

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

sympy.simplify(expr)
expr.simplify()
expr
```

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

[ ]: $x(x-3)$

[ ]: $x(x-3)$

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

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

```
sympy.simplify(expr)
expr = sympy.exp(x) * sympy.exp(y)
expr

sympy.simplify(expr)
```

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

[ ]: $\sin(2x)$

[ ]: $e^x e^y$

[ ]: e^{x+y}

1.5.2 Expand

[ ]:
```
expr = (x + 1) * (x + 2)
sympy.expand(expr)

sympy.sin(x+y).expand(trig=True)

a, b =sympy.symbols("a, b", positive=True)
sympy.log(a * b).expand(log = True)
```

[ ]: $x^2 + 3x + 2$

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

[ ]: $\log(a) + \log(b)$

[ ]:
```
sympy.exp(I*a + b).expand(complex=True)
sympy.expand((a * b)**x, power_base=True)
sympy.exp((a-b)*x).expand(power_exp=True)
```

[ ]: $ie^b\sin(a) + e^b\cos(a)$

[ ]: $a^x b^x$

[ ]: $e^{ax}e^{-bx}$

1.5.3 Factor, Collect, and Combine

[ ]:
```
sympy.factor(x**2 - 1)
sympy.factor(x*sympy.cos(y) + sympy.sin(x) * x)
sympy.logcombine(sympy.log(a) - sympy.log(b))
```

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

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

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

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

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

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

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

1.5.4 Apart, Together, and Cancel

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

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

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

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

1.5.5 Substitution

```
[ ]: (x + y).subs(x, y)
     sympy.sin(x * sympy.exp(x)).subs(x, y)
     sympy.sin(x * z).subs({z: sympy.exp(y), x: y, sympy.sin: sympy.cos})
     expr = x * y + z ** 2 *x
     values = {x: 1.25, y: 0.4, z: 3.2}
     expr.subs(values)
```

[ ]: $2y$

[ ]: $\sin(ye^y)$

[ ]: $\cos(ye^y)$

[ ]: 13.3

1.6 Numerical Evaluation

```
[ ]: sympy.N(1 + pi)
     sympy.N(pi, 50)
     (x + 1/pi).evalf(10)
```

[ ]: 4.14159265358979

[ ]: 3.141592653589793238462643383279502884197169399375l

[ ]: $x + 0.3183098862$

[ ]:
```
expr = sympy.sin(pi * x * sympy.exp(x))
expr_func = sympy.lambdify(x, expr)
expr_func(1.0)
```

[ ]: 0.773942685266709

1.7 Calculus

1.7.1 Derivates

[ ]:
```
f = sympy.Function('f')(x)
sympy.diff(f, x)

sympy.diff(f, x, x)

sympy.diff(f, x, 3)
```

[ ]: $\frac{d}{dx}f(x)$

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

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

[ ]:
```
g = sympy.Function('g')(x, y)
g.diff(x, y)
g.diff(x, 3, y, 2)
```

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

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

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

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

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

1.7.2 Integrals

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

[ ]: $$\int f(x)\,dx$$

[ ]: $$\int\limits_{a}^{b} f(x)\,dx$$

[ ]:
```python
sympy.integrate(sympy.exp(-x**2), (x, 0, oo))
a, b, c = sympy.symbols("a, b, c", positive=True)
sympy.integrate(a * sympy.exp(-((x-b)/c)**2), (x, -oo, oo))
```

[ ]: $$\frac{\sqrt{\pi}}{2}$$

[ ]: $$\sqrt{\pi}ac$$

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

[ ]: $$\int \sin\left(x\cos\left(x\right)\right)dx$$

1.7.3 Series

[ ]:
```python
x, y = sympy.symbols("x, y")
f = sympy.Function("f")(x)
sympy.series(f, x)
x0 = sympy.Symbol("{x_0}")
f.series(x, x0, n=2)
```

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

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

[ ]:
```python
x, y = sympy.symbols("x, y")
f = sympy.Function("f")(x)
sympy.series(f, x)
f.series(x, x0, n=2).removeO()
```

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

[ ]:

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

[ ]:
```
sympy.cos(x).series()
sympy.sin(x).series()
sympy.exp(x).series()
(1/(1+x)).series()
expr = sympy.cos(x) / (1 + sympy.sin(x * y))
expr.series(x, n=4)
expr.series(y, n=4)
```

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

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

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

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

[ ]:
$$1 - xy + x^2\left(y^2 - \frac{1}{2}\right) + x^3\left(-\frac{5y^3}{6} + \frac{y}{2}\right) + O\left(x^4\right)$$

[ ]:
$$\cos(x) - xy\cos(x) + x^2y^2\cos(x) - \frac{5x^3y^3\cos(x)}{6} + O\left(y^4\right)$$

1.7.4 Limits

[ ]:
```
sympy.limit(sympy.sin(x) / x, x, 0)
f = sympy.Function('f')
x, h = sympy.symbols("x, h")
diff_limit = (f(x+h) - f(x))/h
sympy.limit(diff_limit.subs(f, sympy.cos), h, 0)
sympy.limit(diff_limit.subs(f, sympy.sin), h, 0)
```

[ ]:
$$1$$

[ ]:
$$-\sin(x)$$

[ ]:
$$\cos(x)$$

1.7.5 Sums and Products

[ ]:
```
n = sympy.symbols("n", integer=True)
x = sympy.Sum(1/(n**2), (n, 1, oo))
x
x.doit()
x = sympy.Product(n, (n, 1, 7))
x
x.doit()
```

[ ]:

$$\sum_{n=1}^{\infty} \frac{1}{n^2}$$

[ ]:

$$\frac{\pi^2}{6}$$

[ ]:

$$\prod_{n=1}^{7} n$$

[ ]:

$$5040$$

1.7.6 Equations

[ ]:

```
x = sympy.Symbol("x")
sympy.solve(x**2 + 2*x - 3)

a, b, c = sympy.symbols("a, b, c")
sympy.solve(a * x**2 + b * x + c, x)
```

[ ]:

$$[-3, \ 1]$$

[ ]:

$$\left[\frac{-b - \sqrt{-4ac + b^2}}{2a}, \ \frac{-b + \sqrt{-4ac + b^2}}{2a} \right]$$

[ ]:

```
sympy.solve(sympy.sin(x) - sympy.cos(x), x)
sympy.solve(sympy.exp(x) + 2 * x, x)
```

[ ]:

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

[ ]:

$$\left[-W\left(\frac{1}{2}\right) \right]$$

[ ]:

```
eq1 = x + 2 * y - 1
eq2 = x - y + 1
sympy.solve([eq1, eq2], [x, y], dict=True)
eq1 = x **2 - y
eq2 = y**2 - x
sols = sympy.solve([eq1, eq2], [x, y], dict=True)
sols
```

[ ]:

$$\left[\left\{ x : -\frac{1}{3}, \ y : \frac{2}{3} \right\} \right]$$

[ ]:

$$\left[\{x : 0, \ y : 0\}, \ \{x : 1, \ y : 1\}, \ \left\{ x : \left(-\frac{1}{2} - \frac{\sqrt{3}i}{2} \right)^2, \ y : -\frac{1}{2} - \frac{\sqrt{3}i}{2} \right\}, \ \left\{ x : \left(-\frac{1}{2} + \frac{\sqrt{3}i}{2} \right)^2, \ y : -\frac{1}{2} + \frac{\sqrt{3}i}{2} \right\} \right]$$

1.7.7 Linear Algebra

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

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

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

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

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

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

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

M * M
x = sympy.Matrix(sympy.symbols("x_1, x_2"))
M * x
```

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

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

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

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

b = sympy.Matrix(sympy.symbols("b_1, b_2"))
b

x = M.LUsolve(b)
x

x = M.inv() * b
x
```

[ ]: $\begin{bmatrix} 1 & p \\ q & 1 \end{bmatrix}$

[ ]: $\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$

[ ]: $\begin{bmatrix} b_1 - \frac{p(-b_1 q + b_2)}{-pq+1} \\ \frac{-b_1 q + b_2}{-pq+1} \end{bmatrix}$

[ ]: $\begin{bmatrix} \frac{b_1}{-pq+1} - \frac{b_2 p}{-pq+1} \\ -\frac{b_1 q}{-pq+1} + \frac{b_2}{-pq+1} \end{bmatrix}$

Reference * Title: Physics Programming Lecture Note (INU) * Author: Jeongwoo Kim, Ph.D. * Availability: https://sites.google.com/view/jeongwookim

---