Symbolic Python

CS*101* `lec20`

`Symbolic Calculus`

2019-11-27

Announcements

quiz: `quiz20` due on Tues 28/11

lab: `lab10` 29/11

hw: `hw11` due Wed 04/12

Roadmap

Week 1 (9/9)	Week 2 (16/9)	Week 3 (23/9)	30/9
L0 Intro; L1 PY - Intro	L2 PY – Data& Bool; L3 PY – if & Loops	L4 PY - Lists; L5 PY - Funcs	L6 PY - Files
9/10; 12/10	Week 6 (14/10)	Week 7 (21/10)	Week 8 (28/10)
L7 PY - Muta; Exam 01	L8 PY - Dict; L9 PY - IO	L10 PY - Numpy; L11 PY - Plot	L12 PY - State; L13 PY - Rand
Week 9 (4/11)	Week 10 (11/11)	Week 11 (18/11)	Week 12 (25/11)
L14 PY – De-bug; L15 PY - Eqn	L16 Brute; Exam 02	L17 PY - Opt II; L18 PY - Mod	L19 PY – Sym I; L20 PY – Sym II
Week 13 (2/12)	Week 14 (9/12)	Week 15 (16/12)	Week 16 (23/12)
L22 ML - Intro; L23 ML - IO	L24 ML - Poly; L25 ML - Stats	L26 ML - Fit; Re-view	Study Week
Week 17 (30/12)			
Finals			

Objectives

A. Differentiate and integrate expressions.
B. Expand and linearize equations using a Taylor series expansion.
C. Solve simple physics expressions (such as gravity or pendulum swinging) using SymPy.
D. Convert SymPy functions into Python functions.

Review

Question

```
sympy. init_printing()
```

Symbolic Differentiation

sympy.diff

Differentiation is formally calculated:

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

sympy.diff

Differentiation is formally calculated:

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Numerically, this is what we approximate (small Δx).

Symbolically, we apply tables of rules.

sympy.diff

Expression	Derivative Rule
x^n	nx^{n-1}
$\sin x$	$\cos x$
$f(x)g(x)$	$f'(x)g(x) + f(x)g'(x)$
$f(g(x))$	$f'(g(x))g'(x)$
$\frac{f(x)}{g(x)}$	$\frac{f'(x)g(x)-g'(x)f(x)}{g(x)^2}$

Mainly `sympy` mechanically applies these rules.

sympy.diff

Expression	Derivative Rule
x^n	nx^{n-1}

```
import sympy
x,n = sympy.S( 'x,n' )
sympy.diff( x**n,x,1 )
```

The last argument, the order, is optional.

sympy.diff

Expression	Derivative Rule
x^n	nx^{n-1}

```
import sympy
x,n = sympy.S( 'x,n' )
sympy.diff( x**n,x,1 )
```

The last argument, the order, is optional.
Here it is 1 is f ' (x).
2 is f '' (x).

sympy.diff

Expression	Derivative Rule
$\sin x$	$\cos x$

```
import sympy
x = sympy.S( 'x' )
sympy.diff( sympy.cos( x ) + sympy.tan( x ),x,1 )
```

sympy.diff

Expression	Derivative Rule
$f(x)g(x)$	$f'(x)g(x) + f(x)g'(x)$

```
import sympy
x = sympy.S( 'x' )
f = x ** 3
g = sympy.sin( x )
sympy.diff( f * g,x,1 )
```

sympy.diff

Expression	Derivative Rule
$f(g(x))$	$f'(g(x))g'(x)$

```
import sympy
x = sympy.S( 'x' )
g = sympy.sin( 2 * x )
f = g ** 3
sympy.diff( f,x,1 )
```

sympy.diff

Expression	Derivative Rule
$\frac{f(x)}{g(x)}$	$\frac{f'(x)g(x)-g'(x)f(x)}{g(x)^2}$

```
import sympy
x = sympy.S( 'x' )
f = x ** 3
g = sympy.sin( x )
sympy.diff( f / g,x,1 )
```

sympy.diff

Expression	Derivative Rule
$f(x)^{g(x)}$	$f(x)^{g(x)} \left(f'(x) \frac{g(x)}{f(x)} + g'(x) \ln f(x) \right)$

```
import sympy
x = sympy.S( 'x' )
g = sympy.sin( x )
f = x ** 3
sympy.diff( f ** g,x,1 )
```

Symbolic Integration

sympy.integrate

Integration can be definite (bounded) or indefinite.

sympy.integrate

Integration can be definite (bounded) or indefinite.

Indefinite integration involves finding the antiderivative and may fail.

$$\int dx\, x^2 = \frac{x^3}{3} + C$$

`sympy.integrate` drops the constant of integration C.

sympy.integrate

$$\int dx\, x^2 = \frac{x^3}{3} + C$$

```
sympy.integrate( x ** 2,x )
```

sympy.integrate

$$\int dx\, x^y = \begin{cases} \frac{x^{y+1}}{y+1} + C & y \neq 1 \\ \log x + C & y = 1 \end{cases}$$

```
x,y = sympy.S( 'x,y' )
sympy.integrate( x ** y,x )
```

Use this command: `init_printing()` to set the display format.

sympy.integrate

Definite integration requires the bounds as a `tuple`.

$$\int_0^{\frac{\pi}{2}} dx \cos x = 1$$

```
sympy.integrate(sympy.cos( x ),( x,0,sympy.pi/2 ) )
```

sympy.integrate

$$\int_0^1 dx\,\sqrt{x} = \frac{2}{3}$$

```
sympy.integrate( sympy.sqrt( x ),( x,0,1 ) )
```

sympy.integrate

$$\int_0^1 dx\,\sqrt{1+\exp\left(x^2\right)}$$

```
sympy.integrate(sympy.sqrt(1+sympy.exp(-x**2))
                                              ,(x,0,1))
```

If integration unsuccessful, you receive a result with `Integration` sign.

Numerical techniques are typically, then, needed to calculate these results.

sympy.integrate

Multiple integrals can also be handled smoothly.

$$\int_0^1 dy \int_{-1}^{+1} dx \, 2\sin^2 x + 3y$$

```
sympy.integrate(sympy.integrate(2*sympy.sin(x)**2
              +3*y,( x,-1,+1 ) ), ( y,0,1 ) )
```

Taylor Series & Linearization

sympy.series

A Taylor series expands a function by its derivatives at a point.

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

sympy.series

Each term improves on an approximate solution:

Expand at x = 0,

$$\frac{1}{1-x} \approx 1 + x + x^2 + x^3 + \cdots$$

```
sympy.series( 1 / ( 1 - x ), x, 0 )
```

sympy.series

Frequently, only one term in x is taken, which allows method of linear analysis to be applied to a nonlinear function near the selected point.

$$f(x) = \sqrt(x)$$

Using Taylor Series (up to 2nd term):

$$\hat{f}(x)\Big|_{x=a} = \sqrt{a} + \frac{1}{2\sqrt{a}}(x-a)$$

sympy.series

Frequently, only one term in x is taken, which allows method of linear analysis to be applied to a nonlinear function near the selected point.

$$f(x) = \sqrt(x)$$

Using Taylor Series (up to 2nd term):

$$\hat{f}(x)\Big|_{x=a} = \sqrt{a} + \frac{1}{2\sqrt{a}}(x-a)$$

```
f = sympy.sqrt( x )
fhat = sympy.series( f,x,a,2 ).removeO()

.removeO() is a function that removes
higher order terms behind the expansion.

# '2' is the number of terms, not order.
Here '2' contains the constant and x terms
```

Symbolic Physics

Fluid Mechanics Question 1

Barometer

$$p_{\text{atm}} = \gamma h + p_{\text{vapor}}$$

Fluid Mechanics Question 1

Barometer

$$p_{\text{atm}} = \gamma h + p_{\text{vapor}}$$

```
p_atm,g,h,p_vapor = sympy.S( 'p_atm,g,h,p_vapor' )
g_soln = sympy.solve(p_atm-(g*h+p_vapor) ,g)
```

Fluid Mechanics Question 2

Atmospheric pressure by altitude

$$P = P_b \exp\left[-\frac{0.284h}{8.314T_b}\right]$$

Fluid Mechanics Question 2

Atmospheric pressure by altitude

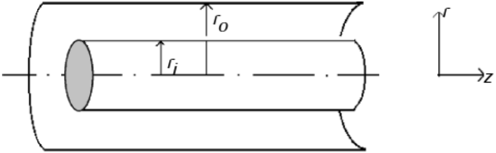
$$P = P_b \exp\left[-\frac{0.284h}{8.314T_b}\right]$$

```
h = sympy.S( 'h' )
P_eqn = 101325 * sympy.exp( -0.284 * h /
                            ( 8.314 * 298.15 ) )
sympy.plotting.plot( P_eqn,( h,0,11000 ) )
```

Fluid Mechanics Question 3

Volumetric rate of flow

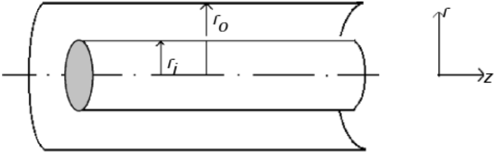
$$Q = \int_{r_i}^{r_o} dr\, v_z (2\pi r)$$



Fluid Mechanics Question 3

Volumetric rate of flow

$$Q = \int_{r_i}^{r_o} dr\, v_z(2\pi r)$$




```
r_i,r_o,r,v_z = sympy.S( 'r_i,r_o,r,v_z' )
Q = sympy.integrate( v_z*2*sympy.pi*r,(r,r_i,r_o))
Q_soln = sympy.simplify( Q )
```

Convert Sympy to regular Py

$$f(a) = (2^{16} - 1)\sqrt[4]{\frac{a}{2^{8} - 1}}$$

Convert Sympy to regular Py

$$f(a) = (2^{16} - 1)\sqrt[4]{\frac{a}{2^8 - 1}}$$

```
a = sympy.S( 'a' )
expr = ( 2 ** 16 - 1 ) *
              sympy.root( a / ( 2 ** 8 - 1 ),4 )
expr.subs(a,10)
```

$$1285\sqrt[4]{2} \cdot 51^{\frac{3}{4}}$$

```
f = sympy.lambdify( [ a ],expr )
```

29163.406987767958

Summary

Summary

A. `sympy.diff(eqn, variable, order)`
B. `sympy.integrate(eqn, variable)` or `sympy.integrate(eqn, (variable, lower, upper))`
C. `sympy.series( eqn, variable, which-Pt, num-terms ).removeO()`
D. Using sympy to represent physics