

SymPy和符号计算

黄书剑



SciPy生态系统



- **SciPy (pronounced “Sigh Pie”) is a Python-based ecosystem of open-source software for mathematics, science, and engineering.**



NumPy

Base N-dimensional
array package



SciPy library

Fundamental library for
scientific computing



Matplotlib

Comprehensive 2-D
plotting

IP[y]:
IPython

IPython

Enhanced interactive
console



SymPy

Symbolic mathematics



pandas

Data structures &
analysis

- 函数、极限、连续
- 一元函数微分学及其应用
- 一元函数积分学及其应用

- 无穷级数
- 多元函数微分学及其应用
- 多元函数积分学及其应用

- 多项式
 - 行列式
 - 线性方程组
 - 矩阵
 - 二次型
-
- 线性空间
 - 线性变换
 - 欧几里得空间
 - 双线性函数与辛空间

- 随机变量及其分布
- 随机变量数字特征
- 集中不等式
- 参数估计
- 假设检验
- 回归分析与方差分析



最优化方法

- 凸集合
- 凸函数
- 凸优化问题
- 对偶性
- 凸函数优化
- 平滑函数优化
- 随机优化
- 分布式优化
- 在线优化

- 一个用于符号化数学计算的工具包

- 更精确的表示数值和运算内容
- 更直接的表示抽象的计算过程
- 用于:

- 函数、极限、连续、积分、微分 (数学分析)
- 多项式、二次型 (高等代数)

符号表示 v.s. 数值表示

pi v.s. 3.14
or 3.141592...3

sqrt(2) v.s. 1.414
or 1.4142135

SymPy:

`exp (x)/(1+exp (2*x))`

$$\frac{e^x}{e^{2x} + 1}$$



SYMPY表达式



SymPy基本元素

- 变量:
 `x = sympy.Symbol("x")`
 `a, b, c = sympy.symbols("a, b, c")`
 ~~`crazy = sympy.symbols('unrelated')`~~

- 数值:
 `i = sympy.Integer(26)`
 `f = sympy.Float(3.8)`
 `r = sympy.Rational(11, 13)`

内建数值:

```
In [49]: sympy.pi
Out[49]:  $\pi$ 
In [50]: sympy.E
Out[50]:  $e$ 
In [51]: sympy.oo
Out[51]:  $\infty$ 
```



SymPy基本元素

- 数值的精确性

```
In [23]: i ** 50
```

```
Out[23]:
```

```
560618465766419330685118611284358470244734599366478937  
58520097689829376
```

```
In [24]: sympy.Float(0.3, 25)
```

```
Out[24]: 0.29999999999999999888977698
```

```
In [25]: sympy.Float('0.3', 25)
```

```
Out[25]: 0.300000000000000000000000000000
```

试着实现一个大整数类，可以进行任意大小的整数运算。
试着实现一个有理数类，可以以分数形式进行运算。
试着实现一个精确浮点数，可以进行精确的加减乘运算。

SymPy基本元素



- 函数:

- 自定义函数 v.s. 内建函数

```
In [52]: f = sympy.Function("f")
...: print(f, type(f))
f <class 'sympy.core.function.UndefinedFunction'>
```

```
In [53]: g = sympy.sin
...: print(g, type(g))
sin <class 'sympy.core.function.FunctionClass'>
```



SymPy基本元素

- 表达式：对计算过程的描述
 - 由变量、数值、函数组成
 - 可以进行函数调用；可以通过sympify创建

```
In [59]: f(i)    # i = sympy.Integer(26)
```

```
Out[59]: f(26)
```

```
In [62]: f(x)    # x = sympy.Symbol("x")
```

```
Out[62]: f(x)
```

```
In [63]: expr = 1 + 2 * x**2 + 3 * x**3
```

```
...: expr
```

```
Out[63]:
```

```
      3      2
3·x  + 2·x  + 1
```

```
In [67]: sympy.sin(sympy.pi)
```

```
Out[67]: 0
```

```
In [68]: sympy.sin(1)
```

```
Out[68]: sin(1)
```

```
In [70]: sympy.simplify("2*x**2 + 3*x**3 + 1")
```

```
In [71]: sympy.simplify("exp(t)/(1 + exp(2 * t))")
```

SYMPY表达式常用功能

表达式操作



- 化简

```
In [72]: expr = 3 * (x ** 2 - x) - x * (x + 1)
```

```
In [73]: expr
```

```
Out[73]:
```

```
      2  
3·x  - x·(x + 1) - 3·x
```

```
In [74]: expr.simplify()
```

```
Out[74]: 2·x·(x - 2)
```

– simplify, trigsimp, powsimp, combsimp, ratsimp 等



表达式操作

- 展开和因式分解

```
In [75]: (x + 1) * (x + 2)
```

```
Out[75]: (x + 1)·(x + 2)
```

```
In [76]: sympy.expand((x + 1) * (x + 2))
```

```
Out[76]:
```

```
      2  
x  + 3·x + 2
```

```
In [77]: expr = 1 + 2 * x**2 + 3 * x**3
```

```
In [78]: sympy.factor(expr)
```

```
Out[78]:
```

```
      2  
(x + 1)· (3·x  - x + 1)
```



表达式操作

- 控制多项式分解

- mul, trig, log, complex, power_base等

```
In [79]: sympy.sin(x + y)
```

```
Out[79]: sin(x + y)
```

```
In [80]: sympy.sin(x + y).expand(trig=True)
```

```
Out[80]: sin(x)·cos(y) + sin(y)·cos(x)
```




表达式操作

• 分式分解、通分和消除公因子

In [81]: `1/ (x**2 + 5*x + 6)`

Out[81]:

$$\frac{1}{x^2 + 5x + 6}$$

In [82]: `sympy.apart(1/ (x**2 + 5*x + 6))`

Out[82]:

$$-\frac{1}{x+3} + \frac{1}{x+2}$$

In [83]: `sympy.together(1/(y*x + x) + 1/(1 + x))`

Out[83]:

$$\frac{x \cdot (y + 1) + x + 1}{x \cdot (x + 1) \cdot (y + 1)}$$

In [84]: `sympy.cancel(x/ (x + x**2))`

Out[84]:

$$\frac{1}{x + 1}$$



表达式操作

- 求值

- sympy.N, evalf

```
In [85]: sympy.N(1 + sympy.pi)
```

```
Out[85]: 4.14159265358979
```

```
In [86]: sympy.N(sympy.pi, 50) #50 significant figures
```

```
Out[86]:
```

```
3.1415926535897932384626433832795028841971693993751
```

```
In [87]: (x + 1/sympy.pi).evalf()
```

```
Out[87]: x + 0.318309886183791
```



表达式操作

- 求值

- 用subs带入变量值(多个值可以使用字典)

```
In [88]: (x + 1/sympy.pi).subs(x, 3).evalf()
```

```
Out[88]: 3.31830988618379
```

```
In [89]: expr = x * y + z**2 * x
```

```
In [90]: values = {x: 1.25, y: 0.4, z: 3.2}
```

```
In [91]: expr.subs(values)
```

```
Out[91]: 13.300000000000000
```



表达式操作

- 对NumPy数组进行矢量化操作

```
In [89]: expr = sympy.cancel(x/ (x + x**2))
```

```
In [90]: expr_func = sympy.lambdify(x, expr, 'numpy')
```

```
In [91]: import numpy as np
```

```
....: xlist = np.arange(0, 10)
```

```
In [92]: %timeit expr_func(xlist)
```

```
2.03 µs ± 58 ns per loop (mean ± std. dev. of 7 runs,  
100000 loops each)
```

```
In [93]: %timeit [expr.subs(x, v).evalf() for v in xlist]
```

```
379 µs ± 2.71 µs per loop (mean ± std. dev. of 7 runs, 1000  
loops each)
```

SYMPY和线性代数



方程组求解

- **solve 函数求解单个方程**

```
In [106]: sympy.solve(x**2 + 2*x - 3)
```

```
Out[106]: [-3, 1]
```

- 通过第二个参数指定需求解的变量

```
In [107]: sympy.solve(sympy.sin(x) - sympy.cos(x), x)
```

```
Out[107]:  
[  
  -3·π   π  
  -----, -  
    4     4  
]
```



方程组求解

- **solve 函数求解单个方程**
 - 解也是符号化的形式

```
[In [94]: sympy.solve(a * x**2 + b * x + c, x)
```

```
Out[94]:
```

$$\left[\frac{-b + \sqrt{-4 \cdot a \cdot c + b^2}}{2 \cdot a}, \frac{-\left(b + \sqrt{-4 \cdot a \cdot c + b^2}\right)}{2 \cdot a} \right]$$



方程组求解

- 求解方程组

- 方程、变元都以列表形式提供

```
In [108]: eq1 = x + 2 * y - 1
...: eq2 = x - y + 1
...:
...: sympy.solve([eq1, eq2], [x, y], dict=True)
Out[108]: [{x: -1/3, y: 2/3}]
```




方程组求解

- 求解方程组

- 方程、变元都以列表形式提供
- 高次方程可能会返回多个解

```
In [109]: eq1 = x**2 - y
...: eq2 = y**2 - x
...:
...: sympy.solve([eq1, eq2], [x, y], dict=True)
```

Out[109]:

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



矩阵及其运算

- 二维矩阵: `sympy.Matrix`
 - 矩阵元素也可以是符号

```
In [104]: sympy.Matrix([1,2])
```

```
Out[104]:  $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ 
```

```
In [105]: sympy.Matrix([[1,2]])
```

```
Out[105]:  $\begin{bmatrix} 1 & 2 \end{bmatrix}$ 
```

```
In [106]: sympy.Matrix([[1, 2], [3, 4]])
```

```
Out[106]:  $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ 
```

```
In [107]: a, b, c, d = sympy.symbols("a, b, c, d")  
M = sympy.Matrix([[a, b], [c, d]])
```

```
In [108]: M
```

```
Out[108]:  $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ 
```

```
In [109]: M * M
```

```
Out[109]:  $\begin{bmatrix} a^2 + bc & ab + bd \\ ac + cd & bc + d^2 \end{bmatrix}$ 
```

矩阵及其运算



- **transpose, adjoint**
- **trace, det, inv**
- **norm, rank, singular_values**
- **solve**
- **.....**

SYMPY和数学分析

- 计算给定表达式在某个给定点的极限

```
In [116]: sympy.limit(sympy.sin(x) / x, x, 0)
```

```
Out[116]: 1
```

- 计算给定表达式在某个给定点的极限

```
In [117]: f = sympy.Function('f')
...: x, h = sympy.symbols("x, h")
...: diff_limit = (f(x + h) - f(x))/h
...: diff_limit
```

```
Out[117]:
-f(x) + f(h + x)
-----
h
```

```
In [119]: sympy.limit(diff_limit.subs(f, sympy.cos), h, 0)
```

```
Out[119]: -sin(x)
```

- 计算给定表达式在某个给定点的极限

- sympy.oo为正无穷

```
In [120]: expr = (x**2 - 3*x) / (2*x - 2)
          ...: expr
```

```
Out[120]:
```

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

```
In [122]: sympy.limit(expr/x, x, sympy.oo)
```

```
Out[122]: 1/2
```

- 求给定级数的极限

```
In [167]: n = sympy.symbols("n", integer=True)
```

```
In [169]: sumlist = sympy.Sum(1/(n**2), (n, 1, sympy.oo))  
sumlist
```

```
Out[169]: 
$$\sum_{n=1}^{\infty} \frac{1}{n^2}$$

```

```
In [170]: sumlist.doit()
```

```
Out[170]: 
$$\frac{\pi^2}{6}$$

```




- 求给定数列的乘积

```
In [172]: productlist = sympy.Product(n, (n, 1, 7))  
productlist
```

```
Out[172]: 
$$\prod_{n=1}^7 n$$

```

```
In [173]: productlist.doit()
```

```
Out[173]: 5040
```



级数展开（泰勒公式）

- 用级数表示函数在给定点的函数值：

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x - a)^n$$

In [192]: `sympy.series(f, x)`

Out[192]:

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

In [195]: `x0 = sympy.Symbol("{x_0}")`
`f.series(x, x0, n=3)`

Out[195]:

$$f(x_0) + (x - x_0) \frac{d}{d\xi_1} f(\xi_1) \Big|_{\xi_1=x_0} + \frac{(x - x_0)^2 \frac{d^2}{d\xi_1^2} f(\xi_1) \Big|_{\xi_1=x_0}}{2} + O((x - x_0)^3; x \rightarrow x_0)$$



级数展开（泰勒公式）

- 用级数表示函数在给定点的函数值

In [197]: `sympy.cos(x).series(n=10)`

Out[197]: $1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + \frac{x^8}{40320} + O(x^{10})$

In [191]: `sympy.series(1/(1-x))`

Out[191]: $1 + x + x^2 + x^3 + x^4 + x^5 + O(x^6)$

In [198]: `sympy.series(sympy.ln(1 + x))`

Out[198]: $x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} + O(x^6)$

导数和微分

- 求给定函数的导数
 - 可多次求导

```
In [189]: f = sympy.Function('f')(x)  
          type(f)
```

```
Out[189]: f
```

```
In [136]: sympy.diff(f, x)
```

```
Out[136]:  $\frac{d}{dx} f(x)$ 
```

```
In [137]: sympy.diff(f, x, x)
```

```
Out[137]:  $\frac{d^2}{dx^2} f(x)$ 
```

```
In [138]: sympy.diff(f, x, 3)
```

```
Out[138]:  $\frac{d^3}{dx^3} f(x)$ 
```



导数和微分

- 求给定函数的导数
 - 可多次求导
 - 可对多元函数逐次求导

```
In [139]: g = sympy.Function('g')(x, y)
```

```
In [140]: g.diff(x, y)
```

```
Out[140]:  $\frac{\partial^2}{\partial y \partial x} g(x, y)$ 
```

```
In [141]: g.diff(x, 3, y, 2)
```

```
Out[141]:  $\frac{\partial^5}{\partial y^2 \partial x^3} g(x, y)$ 
```



导数和微分

- 求具体表达式的导数

```
In [147]: expr = x**4 + x**3 + x**2 + x + 1  
          expr
```

```
Out[147]:  $x^4 + x^3 + x^2 + x + 1$ 
```

```
In [143]: expr.diff(x)
```

```
Out[143]:  $4x^3 + 3x^2 + 2x + 1$ 
```

```
In [145]: expr.diff(x,2)
```

```
Out[145]:  $2(6x^2 + 3x + 1)$ 
```

- 求具体表达式的导数

```
In [148]: expr = (x + 1)**3 * y ** 2 * (z - 1)
          expr
```

```
Out[148]:  $y^2(x + 1)^3(z - 1)$ 
```

```
In [149]: expr.diff(x, y, z)
```

```
Out[149]:  $6y(x + 1)^2$ 
```



微分表达式及其计算

- **sympy.Derivative**
- **doit**

```
In [154]: expr = sympy.exp(sympy.cos(x))  
          expr
```

```
Out[154]:  $e^{\cos(x)}$ 
```

```
In [155]: d = sympy.Derivative(expr, x)  
          d
```

```
Out[155]:  $\frac{d}{dx} e^{\cos(x)}$ 
```

```
In [153]: d.doit()
```

```
Out[153]:  $-e^{\cos(x)} \sin(x)$ 
```


- 不定积分

```
In [157]: sympy.integrate(f)
```

```
Out[157]:  $\int f(x) dx$ 
```

- 定积分

- 通过元组给定变量和积分区间

```
In [158]: sympy.integrate(f, (x, a, b))
```

```
Out[158]:  $\int_a^b f(x) dx$ 
```

- 定积分示例

```
In [126]: sympy.integrate(sympy.sin(x))
```

```
Out[126]: -cos(x)
```

```
In [127]: sympy.integrate(sympy.sin(x), (x, a, b))
```

```
Out[127]: cos(a) - cos(b)
```

```
In [128]: sympy.integrate(sympy.sin(x), (x, 0, sympy.pi))
```

```
Out[128]: 2
```



积分

- 多元函数积分

```
In [163]: expr = (x + y)**2  
          expr
```

```
Out[163]: (x + y)2
```

```
In [166]: sympy.integrate(expr, x)
```

```
Out[166]:  $\frac{x^3}{3} + x^2y + xy^2$ 
```

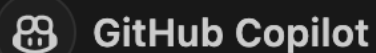
```
In [165]: sympy.integrate(expr, x, y)
```

```
Out[165]:  $\frac{x^3y}{3} + \frac{x^2y^2}{2} + \frac{xy^3}{3}$ 
```

```
In [164]: sympy.integrate(expr, (x, 0, 1), (y, 0, 1))
```

```
Out[164]:  $\frac{7}{6}$ 
```

在SymPy中如何求一个函数的极限



在Python的SymPy库中，你可以使用 `limit` 函数来求解一个函数的极限。以下是一个示例：

```
from sympy import symbols, limit, sin

x = symbols('x')
f = sin(x) / x

lim_f_at_0 = limit(f, x, 0)


print('Limit of f(x) as x approaches 0:', lim_f_at_0)
```

在这个例子中，我们首先定义了一个符号 `x`，然后定义了函数 `f(x) = sin(x) / x`。然后我们使用 `limit` 函数求解了 `f(x)` 在 `x` 趋近于0时的极限。最后，我们打印出了极限的值。

你可以将 `f` 和 `0` 替换为你要求解的函数和点，以求解其他函数的极限。

在SymPy中如何求多元函数的积分



 GitHub Copilot



在Python的SymPy库中，你可以使用 `integrate` 函数来求解多元函数的积分。以下是一个示例：

```
from sympy import symbols, integrate, exp

x, y = symbols('x y')
f = exp(-x**2 - y**2)

integral_f = integrate(f, (x, -oo, oo), (y, -oo, oo))

print('Integral of f(x, y) over all space:', integral_f)
```

在这个例子中，我们首先定义了两个符号 `x` 和 `y`，然后定义了函数 `f(x, y) = exp(-x**2 - y**2)`。然后我们使用 `integrate` 函数求解了 `f(x, y)` 在全空间上的积分。最后，我们打印出了积分的值。

你可以将 `f`，`-oo`，`oo` 替换为你要求解的函数和积分区间，以求解其他多元函数的积分。注意，`-oo` 和 `oo` 在SymPy中分别代表负无穷和正无穷。

- **基本元素**
 - 变量、数值、函数、表达式
- **线性代数**
 - 方程组、矩阵
- **数学分析**
 - 极限、级数、微分、积分
- **更多阅读: <https://docs.sympy.org/latest/tutorial/intro.html>**
- **尝试用SymPy配合相关课程内容学习**
 - 实现部分算法 (求矩阵的特征)
 - 验证相应微分、积分的求解正确性