Python 科学计算基础

第十章 符号计算

2025 年 9 月 5 日

目录

符号计算

符号计算指不使用数值代入变量的情况下按照规则对数学表达式进行精确演算。符号计算和数值计算是两种不同类型的计算。例如要计算函数 $f(x) = x^3 - 2x^2 + 7x - 9$ 的导函数在 $x = 2$ 时的函数值。

- 符号计算根据求导规则获得导函数为 $f'(x) = 3x^2 - 4x + 7$，然后将 $x = 2$ 代入求函数值 $f'(2)$。
- 数值计算则根据数值求导公式对 $x = 2$ 近似计算 $f'(2)$。

虽然很多实际问题无法通过精确演算获得解析解 (例如一元五次方程求根)，符号计算可以用来简化问题的复杂度或缩小问题的规模。

符号计算软件

可进行符号计算的软件称为计算机代数系统 (computer algebra systems)。在众多类似软件中，SymPy 是一个完全用 Python 实现的轻量级开源 Python 库。SageMath 是一个庞大的开源数学软件系统，包含 SymPy 在内的很多开源软件包。Maple 和 Mathematica 则是付费使用的商业软件。

使用 SymPy 模块之前需要运行 “import sympy as sym” 将其导入，使用别名 sym 以便于输入。

符号表达式

符号表达式由符号变量、符号数值、运算符和符号函数按照规则组合而成。

符号变量在使用前需要声明。sym.Symbol 函数可声明单个符号变量，并可用关键字参数指定变量表示的数值的性质。sym.symbols 函数可同时声明多个符号变量。

程序 10.1

符号数值类型

SymPy 的三个类 Integer、Float 和 Rational 分别表示符号数值中的整数、浮点数和有理数，并可通过函数 int 和 float 转换为 Python 中的 int 和 float 类型。SymPy 使用 mpmath 模块，可以进行精确的浮点数计算。SymPy 中定义复数时用 sym.I 表示虚部符号。

程序 10.2

符号函数

符号函数分为三类：sym.Function 声明的未定义的函数、SymPy 内置的常用数学函数和 sym.Lambda 定义的匿名函数。

输入	输出
x, y = sym.symbols("x, y"); f = sym.Function("f")(x, y); f	$f(x, y)$
j = sym.Symbol("j", odd=True); sym.tan(j*sym.pi)	0
k = sym.Symbol("k", even=True); sym.cos(k*sym.pi)	1
sym.sqrt(x*x)	$\sqrt{x^2}$
d = sym.Symbol("d", negative=True); sym.sqrt(d*d)	$-d$
sym.log(sym.E ** sym.Float('-23.12345678987654321'))	-23.12345678987654321
g = sym.Lambda(x, 3 * x**2 - 6 * x + 15); g	$\left(x \mapsto 3x^2 - 6x + 15\right)$
x = sym.Float('3.14'); g(x)	25.7388000000000
sym.simplify(g(x - sym.I * 2))	13.7388 - 25.68*I

simplify

sym.simplify 使用各种经验规则对符号表达式进行化简，适合交互使用，但有时不能得到预期的结果。对于每一类数学函数，SymPy 都定义了专门的化简函数。

输入	输出
x = sym.symbols("x")	
sym.simplify(sym.sin(x)**2 + sym.cos(x)**2)	1
sym.simplify((x**3 - 1)/(x**2 + x + 1))	$x - 1$
expr = sym.exp((x + sym.pi/2) * sym.I); expr	$e^{i\left(x+\frac{\pi}{2}\right)}$
sym.simplify(expr)	ie^{ix}

sympify

sym.sympify 可从一个表示数学表达式的字符串生成一个符号表达式。

输入	输出
x, y, z, u, v, w = sym.symbols("x, y, z, u, v, w")	
expr1 = sym.sympify('sin(x)-tan(x)'); expr1	$\sin(x) - \tan(x)$

subs

subs 函数可将一个符号表达式中出现的所有指定的变量替换为一个指定的表达式，替换的结果保存在一个新的符号表达式中。subs 函数的参数是指定的变量和指定的表达式。如果需要对多个变量替换，可提供一个列表作为参数，列表中的每个元素是由变量和表达式组成的元组。

输入	输出
expr1.subs(x, sym.pi/3)	$-\frac{\sqrt{3}}{2}$
expr = x*y + 2*y*z - 3*z*x + 9; expr	$xy - 3xz + 2yz + 9$
expr2 = expr.subs([(x, 2*u+v), (y, 3*w-u), (z, w)]); expr2	$2w\,(-u + 3w) - 3w\,(2u + v) + (-u + 3w)\,(2u + v) + 9$

evalf

evalf 函数可计算符号表达式的数值。evalf 函数的参数 subs 是一个字典，其中的每个元素由变量和对应的取值组成。evalf 函数的参数 n 指定有效数字的位数，参数 chop 取值为 True 可以去除小于指定精度的舍入误差。

输入	输出
expr2.evalf(subs={u: 2.4, v: sym.pi, w: sym.E})	21.226761448365
expr2.evalf(subs={u: 2.4, v: sym.pi, w: sym.E}, n = 25)	$21.226761448364982075887 46$
expr = sym.cos(sym.E)**2 + sym.sin(sym.E)**2; expr	$\sin^2(e) + \cos^2(e)$
(expr - 1).evalf(), (expr - 1).evalf(chop=True)	(-0.e-125, 0)

lambdify

sym.lambdify 函数可将一个符号表达式(第二个参数)转换成为以指定的变量(第一个参数)作为自变量的 Python 函数以便调用，并通过第三个参数指定表达式中的数学函数由哪个数学库提供。

输入	输出
f = sym.lambdify([u, v, w], expr2)	
f(1,2,3)	53
h = sym.lambdify(x, expr1, 'numpy')	
import numpy as np; h(np.pi/3)	-0.8660254037844382

求导数

sym.diff 可计算单变量函数或多变量函数的导数。它的第一个参数是要求导的函数，后面的一个或多个参数为依次求导的自变量序列。如果对同一个自变量求多次导数，也可以在自变量参数的后面用数字表示求导的次数。sym.Derivative 生成导数的未求值形式，可调用 doit 函数计算导数。

输入	输出
expr = x**4 * sym.E**x; expr	$x^4 e^x$
sym.diff(expr, x, 3)	$x\left(x^3+12x^2+36x+24\right)e^x$
sym.diff(expr, x, x, x)	$x\left(x^3+12x^2+36x+24\right)e^x$
expr = sym.E**(x**2 * y**3) * (2 - x - z); expr	$\left(-x-z+2\right)e^{x^2y^3}$
sym.diff(expr, y, 2, x, z)	$-6xy\left(3x^4y^6+8x^2y^3+2\right)e^{x^2y^3}$
deriv = sym.Derivative(expr, z, y, 3, x); deriv	$\frac{\partial^5}{\partial x\partial y^3\partial z}\left(-x-z+2\right)e^{x^2y^3}$
deriv.doit()	$-6x\left(9x^6y^9+45x^4y^6+38x^2y^3+2\right)e^{x^2y^3}$

求积分

sym.integrate 可计算单变量或多变量函数的定积分和不定积分。计算定积分需要提供自变量的积分范围。sym.Integral 生成积分的未求值形式，可调用 doit 函数计算积分。sym.oo 表示正无穷。对于无法计算的积分，sympy 返回其未求值形式。

输入	输出
sym.integrate(sym.tan(x), x)	$-\log(\cos(x))$
sym.integrate(sym.cos(x**2), x)	$\frac{\sqrt{2}\sqrt{\pi}C\left(\frac{\sqrt{2}x}{\sqrt{\pi}}\right)\Gamma\left(\frac{1}{4}\right)}{8\Gamma\left(\frac{5}{4}\right)}$
sym.integrate(x**x, x)	$\int x^x\,dx$
expr = sym.Integral(sym.exp(-x**2), (x, 0, sym.oo)); expr	$\int_0^\infty e^{-x^2}\,dx$
expr.doit()	$\frac{\sqrt{\pi}}{2}$
expr = sym.Integral(x*sym.log(x), x); expr	$\int x\log(x)\,dx$
expr.doit()	$\frac{x^2\log(x)}{2}-\frac{x^2}{4}$

求积分

输入	输出
expr = sym.Integral(sym.exp(-x**2 - y**4), (x, -sym.oo, 0), (y, 0, sym.oo)); expr	$\int_0^{\infty}\int_{-\infty}^{0} e^{-x^2-y^4}\,dx\,dy$
expr.doit()	$\frac{\sqrt{\pi}\Gamma\left(\frac{1}{4}\right)}{8}$
_.evalf(20)	0.80327828046363949031
expr = sym.Integral(x**y*sym.exp(-x*x), (x, 0, sym.oo)); expr	$\int_0^{\infty} x^y e^{-x^2}\,dx$
expr.doit()	$\begin{cases} \frac{\Gamma\left(\frac{y}{2}+\frac{1}{2}\right)}{2} & \text{for } \frac{\operatorname{re}(y)}{2}-\frac{1}{2}>-1 \\ \int_0^{\infty} x^y e^{-x^2}\,dx & \text{otherwise} \end{cases}$

求极限

sym.limit(f(x), x, x0) 计算极限 $\lim_{x\to x0} f(x)$。要计算单侧极限，添加第四个参数'+' 或'-'。

sym.Limit 生成极限的未求值形式，可调用 doit 函数计算极限。

输入	输出
sym.limit(sym.tan(x)/x, x, 0)	1
sym.limit(x**3/1.03**x, x, sym.oo)	0
expr = sym.Limit(x*(x-2)/sym.Abs(x-2), x, 2, '+'); expr	$\lim_{x\to 2+}\left(\frac{x(x-2)}{\lvert x-2\rvert}\right)$
expr.doit()	2
expr = sym.Limit(x*(x-2)/sym.Abs(x-2), x, 2, '-'); expr	$\lim_{x\to 2-}\left(\frac{x(x-2)}{\lvert x-2\rvert}\right)$
expr.doit()	-2

泰勒级数展开

series 函数可将一个函数在指定的自变量值(第一个参数)处进行泰勒级数展开到指定的阶数(第二个参数)。removeO 函数可去除 series 函数的运行结果中的高阶无穷小项。

In[1]:	expr = sym.tan(x); expr.series(x, 2, 6)
Out[1]:	$\tan(2) + \left(1+\tan^2(2)\right)(x-2) + (x-2)^2\left(\tan^3(2)+\tan(2)\right) + (x-2)^3\left(\frac{1}{3}+\frac{4\tan^2(2)}{3}+\tan^4(2)\right) + (x-2)^4\left(\tan^5(2)+\frac{5\tan^3(2)}{3}+\frac{2\tan(2)}{3}\right) + (x-2)^5\left(\frac{2}{15}+\frac{17\tan^2(2)}{15}+2\tan^4(2)+\tan^6(2)\right) + O\left((x-2)^6; x\to 2\right)$
In[2]:	expr.series(x, 2, 6).removeO()
Out[2]:	$(x-2)^5\left(\frac{2}{15}+\frac{17\tan^2(2)}{15}+2\tan^4(2)+\tan^6(2)\right) + (x-2)^4\left(\tan^5(2)+\frac{5\tan^3(2)}{3}+\frac{2\tan(2)}{3}\right) + (x-2)^3\left(\frac{1}{3}+\frac{4\tan^2(2)}{3}+\tan^4(2)\right) + (x-2)^2\left(\tan^3(2)+\tan(2)\right) + \left(1+\tan^2(2)\right)(x-2) + \tan(2)$

创建矩阵

SymPy.Matrix 表示一个矩阵。创建一个矩阵的方式有多种。

- 给定一个嵌套列表，SymPy.Matrix 生成一个矩阵，嵌套列表中的每个子列表构成一个行向量。
- 给定一个非嵌套列表，SymPy.Matrix 生成一个列向量。
- eye(n) 生成一个 $n \times n$ 的单位阵。
- zeros(n, m) 生成一个 $n \times m$ 的所有元素为 0 的矩阵。
- ones(n, m) 生成一个 $n \times m$ 的所有元素为 1 的矩阵。 diag 生成一个对角阵，它的参数可以是数值或矩阵，它们沿对角方向堆叠。

创建矩阵

输入	输出
sym.Matrix([[1, 2], [-3, -4], [5, 6]])	$\begin{bmatrix} 1 & 2 \\ -3 & -4 \\ 5 & 6 \end{bmatrix}$
sym.Matrix([7, -8, 9])	$\begin{bmatrix} 7 \\ -8 \\ 9 \end{bmatrix}$
sym.eye(3)	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
sym.zeros(2, 3)	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
sym.ones(3, 2)	$\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$

创建矩阵

输入	输出
sym.diag(2, -3, 6)	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 6 \end{bmatrix}$
sym.diag(sym.Matrix([7, -8, 9]), 24, sym.ones(2, 2))	$\begin{bmatrix} 7 & 0 & 0 & 0 \\ -8 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 \\ 0 & 24 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$

矩阵的运算

sym.shape 函数返回一个矩阵的 shape 属性，即由行数和列数组成的元组。row 函数可获取指定的行。col 函数可获取指定的列。 row_del 可删除指定的行。 col_del 可删除指定的列。 row_insert 可在指定的位置插入一行并生成一个新矩阵。 col_insert 可在指定的位置插入一列并生成一个新矩阵。

输入	输出
m = sym.Matrix([[1, 2], [-3, -4], [5, 6]]); m	$\begin{bmatrix} 1 & 2 \\ -3 & -4 \\ 5 & 6 \end{bmatrix}$
sym.shape(m), m.shape	((3, 2), (3, 2))

矩阵的运算

输入	输出
m.row(1)	$\begin{bmatrix} -3 & -4 \end{bmatrix}$
m.col(-2)	$\begin{bmatrix} 1 \\ -3 \\ 5 \end{bmatrix}$
m.row_del(1); m	$\begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix}$
m.col_del(-2); m	$\begin{bmatrix} 2 \\ 6 \end{bmatrix}$
m = m.row_insert(1, sym.Matrix([[7]])); m	$\begin{bmatrix} 2 \\ 7 \\ 6 \end{bmatrix}$
m = m.col_insert(-2, sym.Matrix([8, -9, 10])); m	$\begin{bmatrix} 8 & 2 \\ -9 & 7 \\ 10 & 6 \end{bmatrix}$

矩阵的运算

单个矩阵可进行转置和乘方运算。两个矩阵可进行加法、减法和乘法运算。

输入	输出
p = sym.Matrix([[6, -5], [4, -3], [1, 2]]); p	$\begin{bmatrix} 6 & -5 \\ 4 & -3 \\ 1 & 2 \end{bmatrix}$
m+p	$\begin{bmatrix} 14 & -3 \\ -5 & 4 \\ 11 & 8 \end{bmatrix}$
m-p	$\begin{bmatrix} 2 & 7 \\ -13 & 10 \\ 9 & 4 \end{bmatrix}$
q = p.T; q	$\begin{bmatrix} 6 & 4 & 1 \\ -5 & -3 & 2 \end{bmatrix}$
s = m * q; s	$\begin{bmatrix} 38 & 26 & 12 \\ -89 & -57 & 5 \\ 30 & -22 & 22 \end{bmatrix}$

矩阵的运算

det 函数可获取矩阵的行列式。rref 函数将一个矩阵转换成简化行阶梯形式 (reduced row echelon form)，返回值是一个元组，由转换结果和一个存储了主元列的索引值的元组构成。

输入	输出
t = s - 3 * sym.ones(3, 3); t	$\begin{bmatrix} 35 & 23 & 9 \\ -92 & -60 & 2 \\ 27 & 19 & 19 \end{bmatrix}$
s.det(), t.det()	(0, -936)
t ** (-1)	$\begin{bmatrix} \frac{589}{468} & \frac{133}{468} & -\frac{293}{468} \\ -\frac{901}{468} & -\frac{211}{468} & \frac{449}{468} \\ \frac{16}{117} & \frac{11}{234} & -\frac{2}{117} \end{bmatrix}$
t ** 3	$\begin{bmatrix} 28852 & 19444 & 3468 \\ -82204 & -54468 & 6940 \\ 21720 & 15008 & 9272 \end{bmatrix}$
s ** (-1)	NonInvertibleMatrixError: Matrix det == 0; not invertible.

矩阵的运算

rref 函数将一个矩阵转换成简化行阶梯形式 (reduced row echelon form)，返回值是一个元组，由转换结果和一个存储了主元列的索引值的元组构成。nullspace 函数返回零空间的基向量。columnspace 函数返回列空间的基向量。

输入	输出
v = s.rref(); v[0]	$\begin{bmatrix} 1 & 0 & -\frac{11}{2} \\ 0 & 1 & \frac{17}{2} \\ 0 & 0 & 0 \end{bmatrix}$
v[1]	$(0, 1)$
w = sym.Matrix([[1, -2, 3, 4], [8, 6, 4, 9]]);	$\begin{bmatrix} 1 & -2 & 3 & 0 & 0 \\ 8 & 6 & 0 & 0 & 9 \end{bmatrix}$
w.nullspace()	[Matrix([[-13/11], [10/11], [1], [0]]), Matrix([[-21/11], [23/22], [0], [1]])]
w.columnspace()	[Matrix([[1], [8]]), Matrix([[-2], [6]])]

矩阵的运算

eigenvals 函数返回一个字典，其中每个元素由一个特征值和其对应的代数重数组成。 eigenvects 返回一个列表，其中的每个元素是一个元组，由一个特征值、对应的代数重数和特征向量的列表组成。

输入	输出
x = sym.Matrix([[-1, 0, 1, 2], [0, 1, 0, 1], [-1, -4, 1, -2], [0, 1, 0, 1]]); x	$\begin{bmatrix} -1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 1 \\ -1 & -4 & 1 & -2 \\ 0 & 1 & 0 & 1 \end{bmatrix}$
x.eigenvals()	{2: 1, 0: 3}
x.eigenvects()	[(0, 3, [Matrix([[1], [0], [1], [0]]), Matrix([[2], [-1], [0], [1]])]), (2, 1, [Matrix([[-1], [1], [-5], [1]])])]

矩阵的运算

charpoly 函数返回一个以指定变量为自变量的特征多项式。

输入	输出
lamda = sym.symbols('lamda')	λ
p = x.charpoly(lamda); p	$\mathrm{PurePoly}\left(\lambda^4 - 2\lambda^3, \lambda, \mathit{domain} = \mathbb{Z}\right)$
sym.factor(p.as_expr())	$\lambda^3\left(\lambda - 2\right)$

线性方程组求解

linsolve 函数求解线性方程组。可以使用多种方式描述方程组：方程的列表、增广矩阵和
$Ax = b$ 形式。以求解以下线性方程组为例。

$$\begin{bmatrix} 1 & -2 & 3 \\ 4 & -5 & 6 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 4 \\ 8 \end{bmatrix} \tag{1}$$

线性方程组求解

输入	输出
sym.linsolve([x - 2*y + 3*z - 4, 5*x - 6*y + 7*z - 8], (x, y, z))	$\{(z-2,\ 2z-3,\ z)\}$
sym.linsolve(sym.Matrix(([1, -2, 3, 4], [5, -6, 7, 8])), (x, y, z))	$\{(z-2,\ 2z-3,\ z)\}$
m = sym.Matrix(([1, -2, 3, 4], [5, -6, 7, 8])) system = A, b = m[:, :-1], m[:, -1] sym.linsolve(system, x, y, z)	$\{(z-2,\ 2z-3,\ z)\}$

非线性方程求解

solveset 函数对于给定的变量 (第二个参数) 求解非线性方程 (第一个参数)，可通过 domain 参数指定根的取值范围。如果没有解，返回空集。如果无法求解，返回一个条件集合。对于一元多项式方程，sym.roots 返回一个字典，其中的每个元素包括一个根和其对应的重数。

非线性方程求解

输入	输出
sym.solveset(x**2 - 2*x + 2, x)	$\{1 - i, 1 + i\}$
sym.solveset(x**2 - 2*x + 2, x, domain=sym.S.Reals)	$\emptyset$
sym.solveset(sym.tan(x) - 1, x)	$\left\{2n\pi + \frac{5\pi}{4} \mid n \in \mathbb{Z}\right\} \cup \left\{2n\pi + \frac{\pi}{4} \mid n \in \mathbb{Z}\right\}$
sym.solveset(sym.sin(x) - x, x)	$\{x \mid x \in \mathbb{C} \wedge -x + \sin(x) = 0\}$
sym.solveset(x**5 - 4*x**4 + 14*x**2 - 17*x + 6)	$\{-2, 1, 3\}$
sym.roots(x**5 - 4*x**4 + 14*x**2 - 17*x + 6)	{3: 1, -2: 1, 1: 3}

非线性方程组求解

nonlinsolve 函数可求解规模较小的非线性方程组，它的第一个参数是组成非线性方程组的各方程的列表，第二个参数是要求解的变量的列表。

In[1]:	sym.nonlinsolve([x*x + y*y - 2, x*y + 1], x, y)
Out[1]:	$\{(-1,\ 1), (1,\ -1)\}$
In[2]:	sym.nonlinsolve([x*x + y*y + 2, x*y + 1], x, y)
Out[2]	$\{(-i,\ -i), (i,\ i)\}$
In[3]:	eqns = [x**2 - 3*y**2 - 4, x*y + 2]; vars = [x, y]; sym.nonlinsolve(eqns, vars)
Out[3]:	$\left\{\left(-\sqrt{6},\ \frac{\sqrt{6}}{3}\right), \left(\sqrt{6},\ -\frac{\sqrt{6}}{3}\right), \left(-\sqrt{2}i,\ -\sqrt{2}i\right), \left(\sqrt{2}i,\ \sqrt{2}i\right)\right\}$
In[4]:	eqns = [sym.exp(x) + sym.cos(y), y*y - 3*y + 2]; sym.nonlinsolve(eqns, vars)
Out[4]:	$\{(\{i\,(2n\pi + \pi) + \log(\cos(1)) \mid n \in \mathbb{Z}\},\ 1), (\{2ni\pi + \log(-\cos(2)) \mid n \in \mathbb{Z}\},\ 2)\}$
In[5]:	eqns = [x*x + y*y, z*z + y - 1]; vars = [x, y]; sym.nonlinsolve(eqns, vars)
Out[5]:	$\{(-i\,(z-1)\,(z+1),\ -(z-1)\,(z+1)), (i\,(z-1)\,(z+1),\ -(z-1)\,(z+1))\}$

常微分方程求解：牛顿冷却定律

牛顿冷却定律 (Newton's law of cooling) 描述了温度高于周围环境的物体向环境传递热量逐渐冷却时所遵循的规律，即物体温度降低的速率正比于物体和环境的温差：

$$\frac{dT(t)}{dt} = -K(T(t) - T_s)$$

公式中的 $T(t)$ 为要求解的物体温度随时间变化的函数，K 是一个常数，T_s 是环境温度。

常微分方程求解：牛顿冷却定律

dsolve 函数可求解常微分方程，其参数为方程的 SymPy 表示，方程中要求解的函数用 sym.Function 声明。求解这一方程的结果包括一个积分常数 C_1，需要通过代入初值条件 (物体的初始温度 $T(0) = Tb$) 确定。

输入	输出
t, K, Tb, Ts = sym.symbols("t, K, Tb, Ts")	
T = sym.Function("T")	
ode = T(t).diff(t) + K*(T(t) - Ts); sym.Eq(ode, 0)	$K\left(-Ts + T(t)\right) + \frac{d}{dt}T(t) = 0$
ode_sol = sym.dsolve(ode); ode_sol	$T(t) = C_1 e^{-Kt} + Ts$
ics = {T(0): Tb}	
C_eq = ode_sol.subs(t, 0).subs(ics); C_eq	$Tb = C_1 + Ts$
C = sym.solve(C_eq); C	$[C1 : Tb - Ts]$
ode_sol.subs(C[0])	$T(t) = Ts + (Tb - Ts)\, e^{-Kt}$

常微分方程求解：阻尼谐振子

由于阻尼作用 (例如空气阻力)，阻尼谐振子的振幅不断缩小。设阻尼力为 $-b\frac{dx}{dt}$，则描述阻尼谐振子的位移 x 随时间 t 变化的常微分方程为

$$m\frac{d^2}{dt^2}x(t) + b\frac{dx}{dt}x(t) + kx(t) = 0$$

其中 m、b 和 k 均为常数。设初始条件为 $x(0) = 1$ 和 $x'(0) = 0$，程序 10.3 求解方程得到的结果是

$$x(t) = \left(-\frac{b}{2\sqrt{b^2-4km}} + \frac{1}{2}\right)e^{\frac{t\left(-b-\sqrt{b^2-4km}\right)}{2m}} + \left(\frac{b}{2\sqrt{b^2-4km}} + \frac{1}{2}\right)e^{\frac{t\left(-b+\sqrt{b^2-4km}\right)}{2m}}$$

第 4 行至第 15 行的 solve_for_constants 函数自动将所有初值条件代入微分方程的通解以便确定所有积分常数，然后返回特解。

常微分方程求解：阻尼谐振子

为了直观显示 $x(t)$ 对于 m、b 和 k 的依赖关系，固定 $k = 9$ 和 $m = 1$，选取几个不同的 b 值绘制 $x(t)$ 随 t 变化的函数图像。

- 当 $b^2 < 4km$ 即 $b < 6.0$ 时：振荡频率接近于无阻尼时的振荡频率 $\sqrt{\frac{k}{m}}$ 且振荡幅度随时间逐渐变小；比较不同的 b 值在同一时刻的振荡幅度，振荡幅度随 b 值增大而变小。
- 当 $b^2 = 4km$ 时，即 $b = 6.0$ 时，不发生振荡。此时不能直接将 b 的值代入 $x(t)$，而是需要求极限。

实验 10： SymPy 库简介

本实验的目的是掌握使用 SymPy 库进行矩阵计算和方程求解的方法。

在 Blackboard 系统提交一个文本文件 (txt 后缀)，文件中记录每道题的源程序和运行结果。

1. 矩阵计算

计算矩阵 $\begin{bmatrix} 3 & -2 & a & a \\ 0 & 2 & b & b \\ -5 & 2 & -9 & -7 \\ a & -2 & a & 3 \end{bmatrix}$ 的逆矩阵、特征值和特征向量。

其中 a 和 b 都是符号变量。

2. 求解线性方程组

求解以下线性方程组

$$\begin{pmatrix} 3 & -2 & a & a \\ 0 & 2 & b & b \\ -5 & 2 & -9 & -7 \\ a & -2 & a & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} 1 \\ -2 \\ 3 \\ c \end{pmatrix}$$

其中 a、b 和 c 都是符号变量。

3. 求解常微分方程

以初值条件 $f(0) = 16$ 和 $f'(0) = 18$ 求解常微分方程 $f''(x) - 2f'(x) + 3f(x) - 4 = 0$。