CÁLCULO SIMBÓLICO

CO

MÓDULO SYMPY

Ana María Prieto López (1942-2018)

- Primeira programadora en España, de Santiago
- Dende os 21 anos traballou en IBM e foi a primeira programadora en España da empresa informática Bull
- Programou en COBOL e código máquina
- Traballou dende 1969 na Caixa de Aforros de Santiago

Sympy

Máis información en: http://www.sympy.org/en/index.html

- SYMPY é o paquete para o cálculo simbólico.
- Declaración de variables simbólicas (é obrigatorio):
  - **`x = symbols('x')`** : declara x como unha variable simbólica.
- Sympy utiliza obxetos de sympy nas súas operacións (pode comprobarse coa función **`type()`**). Exemplo:
  - `import sympy`
  - x = sympy.simbols('x')
  - type(x)
  - type(3*x+1)

Sympy: métodos dos obxetos sympy (I)

- **`subs(symbol, valor)`** : método dos obxetos de sympy, que substituye “valor” en “symbol” (valor pode ser unha expresión). Exemplos:
  - from sympy import *
  - x,y,z=symbols('x y z')
  - f=x**2-3 # define f(x)
  - f.subs(x, 1) # calcula f(1)=-2
  - f.subs({x:1}) # calcula f(1)= -2
  - f.subs(x, sin(y)) # calcula f(sin(y))= sin(y)**2-3
  - g=2*x-3*y+4*z # define g(x,y,z)
  - g.subs([(x,1), (y,3), (z,2)]) # calcula g(1,3,2)=1

Sympy: métodos dos obxetos sympy (II)

- **`evalf()`** : forza a conversión a un valor numérico do obxeto sympy. Exemplo:
  - x=symbols('x')
  - a=integrate(x,(x,0,1)) ------> devolve 1/2
  - type(a) ----------> sympy.core.numbers.Half
  - a.evalf() ----------> devolve 0.5
- **`evalf(n)`** : aproxima o valor real con n decimais:
  - pi.evalf() -------> devolve 3.14159265358979
  - pi.evalf(30)-----> devolve 3.14159265358979323846264338328
- **`sqrt(x)`**: calcula a raiz cadrada simbólica.
  - sympy.sqrt(8) -------> devolve 2*sqrt(2)

Sympy: métodos dos obxetos sympy (III)

- **`expr.coeff(x, n)`** : devolve o coeficiente de x^n na expresión expr. Exemplo:
  - x=symbols('x')
  - expr = 3 * x**3 + 7* x**2 + 4
  - expr.coeff(x, 2) ----------> devolve 7

Sympy: outras funcións

- **`sympify(expresión) :`** convirte unha expresión(cadea de caracteres) nun obxeto de SymPy.
- **`lambdify(symbols, expresión)`** : convirte unha expresión ou cadea de caracteres nunha función de tipo **lambda**.
- **`factorial(n) :`** factorial do número n.
- **`binomial(n, k)`** : el número de formas para elegir k elementos a partir de un conjunto de n elementos distintos.

Sympy: manipulación de expresións

- **`symplify(expresión) :`** simplifica unha expresión matemática.
- **`factor(expresión)`** : factoriza unha expresión polinomial en factores irreducibles.
- **`expand(expresión)`** : expande unha expresión polinomial a súa forma canónica dunha suma de monomios. Para polinomios, **`factor()`** é o contrario de **`expand()`**. As funcións **`expand_trig(expresion)`** e **`expand_log(expresion)`**realiza unha expansión utilizando as regras trigonométricas e logarítmicas respectivamente.
- **`collect(expresión)`** : agrupa termos con igual potencia dentro da expresión.

Sympy: matrices (I)

- As matrices en sympy representanse por un obxeto de tipo **`Matrix`**. Exemplos:
  - `M = Matrix([[1, -1, 2], [3, 4, 1]])` # crea unha matriz de 2x3
  - `N = Matrix([1, 2, 3])` # crea unha matriz columna
  - `M*N` # multiplica matrices
  - `M.shape` # devolve o tamaño matriz (2, 3)
  - `M.row(0)` # devolve primeira fila
  - `M.col(-1)` # devolve a última columna
  - `C = M.row_insert(1, Matrix([[3,6,9]]))` # devolve matriz con fila insertada na posición 1.

Sympy: matrices (II)

- `D = M.col_insert(0, Matrix([7,4]))` # devolve coa columna insertada na posición 0.
- `M.col_del(0)` # borra columna 0 da matriz M
- `M.row_del(-1)` # borra última fila da matriz
- `M = Matrix([[1, -1], [3, 4]])` # crea unha matriz de 2x2
- `N = Matrix([[1, 2],[5, 3]])`
- `M+N` # sumaa matrices
- `M-N` # resta matrices
- `M**2` # potencia de matrices
- `M**-1` # inversa dunha matriz
- M.T # transposta dunha matriz

Sympy: matrices (III)

- `M.det()` # devolve determinante da matriz M
- `eyes(4)` # crea matriz identidade de orde 4.
- `zeros(2,3)` # crea matriz con ceros de orde 2x3
- `ones(3,4)` # crea matriz con uns de orde 3x4
- `diag(1, 2, 3)` # crea unha matriz de orde 3 con estos elementos na diagonal
- `M = Matrix([[3, -2, 4, -2], [5, 3, -3, -2], [5, -2, 2, -2], [5, -2, -3, 3]])`
- `M.eigenvals()` # devolve os autovalores ou valores propios de M
- `M.eigenvects()` # devolve os autovectores ou vectores propios de M
- `P, D = M.diagonalize()` # en D devolve a matriz M diagonalizada

Sympy: funcións de análise matemático (I)

- **`limit(f, var, p)`** : calcula o límite dunha función f nun punto p con respecto a unha variable var. **`Limit(f, var, p)`** visualiza o límite.
- **`diff(f, var, n)`** : calcula a derivada de orde n dunha función f respecto a unha variable var. **`Derivative(f, var, n)`** visualiza a derivada.
- **`integrate(f, var)`** : calcula a integral indefinida da función f respecto a variable var. **`Integral(f, var)`** visualiza a integral.
- **`integrate(f, (var, a, b))`** : calcula a integral definida de f respecto a var no intervalo [a,b].

Sympy: funcións de análise matemático (II)

- Calcula a expansión en serie da función f(x) no punto $x=x_0$ con n termos usando **`f(x).series(x,x0,n)`** (por defecto se se omiten os argumentos utiliza x=0 e n=6). Exemplo:
  - `expr = sin(x)`
  - `expr.series(x, 3, 7)`
- **`solve(f, *symbols, **flags)`** : resolve simbólicamente ecuacións (calcula o x tal que f(x)=0) ou sistemas de ecuacións. Se devolve [ ] quere decir que python non atopou solución.
- **`roots(p)`** : calcula as raíces do polinomio p
- **`dsolve()`** : para resolver ecuacións diferencias.

Scipy.optimize

- **`newton(func, x0, fprime=None, args=(), tol=1.48e-08, maxiter=50, fprime2=None)`**: atopa o valor no que func(x)=0 a partir dun punto inicial x0. Utiliza o método de Newton-Raphson se introduces a derivada primeira fprime, senon utiliza o método da secante. Se introduces a derivada segunda fprime2, utiliza o método parabólico de Halley. Tol é a tolerancia e maxiter o número máximo de iteracións.