Mokslinių skaičiavimų bibliotekos

Albertas Gimbutas

[1]Matematikos ir informatikos institutas
Vilniaus universitetas

2014 pavasaris

Turinys

- **SciPy** - tai Python kalba grįsta ekosistema atviro kodo matematikos, mokslo ir inžinerijos programinei įrangai. Ši ekosistema susideda iš **NumPy**, **SciPy**, **Matplotlib**, **IPython**, **Sympy** ir **pandas** modulių.
- **SciPy** - modulis yra rinkinys paketų, sprendžiančių skirtingas mokslinių skaičiavimų problemas, pvz.:
  - **scipy.integrate** - integralų ir diferencialinių lygčių sprendimui
  - **scipy.linalg** - tiesinės algebros ir matricų dekompozicijos paketas (papildantis **numpy.linalg** paketą).
  - **scipy.optimize** - optimizavimo ir sprendimo paieškos algoritmai
  - **scipy.signal** - signalų apdorojimo įrankis
  - **scipy.stats** - tolydžiųjų ir diskrečijų skirstinių (), statistiniai testai.
  - **scipy.weave** - įrankis C++ kodui įterpti efektyvesniams skaičiavimams.

- www.scipy.org
- Book: Python for Data Analysis

- **NumPy** (Numerical Python) - esminė mokslinių skaičiavimų biblioteka, įgalinanti greitus, efektyvius veiksmus su matricomis, tiesinės algebros operacijas ir kt.
- **pandas** - suteikia turtingas duomenų struktūras ir funkcijas lengvam ir efektyviam darbui su struktūruotais duomenimis. Pagrinde dirbama su DataFrame objektu, kuris yra dviejų dimensijų tabais atskirta, į stulpelius orientuota duomenų struktūra su stulpelių ir eilučių pavadinimais.
- **matplotlib** - populiariausias grafikų piešimo ir duomenų vizualizavimo įrankis. Vizualizuoti galima tiek 2D, tiek 3D erdvę.
- **SymPy** - simbolinės matematikos paketas.
- **IPython** - praplėsta Python interaktyvi aplinka, galinti turėti grafinę sąsają. Šioje aplinkoje galima patogiau vykdyti, testuoti Python kodą.

NumPy

```
>>> from numpy import array, mat
>>> a = array([[1, 2], [3, 4], [5, 6]])
>>> m = mat('1 2; 3 4; 5 6')
>>> a.size
6
>>> a.shape
(3, 2)
>>> a[1,:]
array([3, 4])
>>> array([[1, 2, 3]], dtype=complex)
array([[ 1.+0.j,  2.+0.j,  3.+0.j]])
>>> a.T
>>> a.dot(a.T)
```

- Numpy tutorial
- Numpy for Matlab users
- DSP realizacija

Pandas

- Pandas tutorials

Matplotlib

```
>>> import matplotlib.pyplot as plt
>>> plt.plot([0, 0.5, 1], [0, 1, 2], 'go')
>>> plt.plot([0, 0.7, 0.8], [0, 1, 2], 'y*')
>>> plt.show()
```

```
def draw3d(points):
    points = np.array(points)
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(points[:,0], points[:,1], points[:,2])
    plt.show()
```

- Matplotlib docs

SymPy

```
>>> from sympy import *
>>> init_printing(use_latex=True)
```

```
>>> x = symbols('x')
>>> expr = x + 1
>>> expr.subs(x, 2)
3
>>> expr = cost(x) + 1
>>> expr.subs(x, x**y)
>>> expr.subs(x, x**x)
x**(x**x) + 1
>>> expr.subs(x**x, y)
>>> expr.subs([(x**x, y), (x, z)])
```

SymPy

```
>>> a = (x + 1)**2
>>> b = x**2 + 2*x + 1
>>> simplify(a - b)
0
>>> c = x**2 - 2*x + 1
>>> simplify(a - c)
4*x
```

SymPy galimybių pavyzdžiai

```
>>> diff(cos(x), x)
-sin(x)
>>> integrate(exp(-x), (x, 0, oo))
1
>>> exp = exp(-x)
 -x
 e
>>> exp.integrate((x, 0, oo))
1
>>> Integral(exp, (x, 0, oo))
```

```
>>> limit(sin(x)/x, x, 0)
1
>>> Limit(sin(x)/x, x, 0)
>>> solve(x**2 - x, x)
>>> M = Matrix([[1, 0, 1], [2, -1, 3], [4, 3, 2]])
>>> M.det()
```