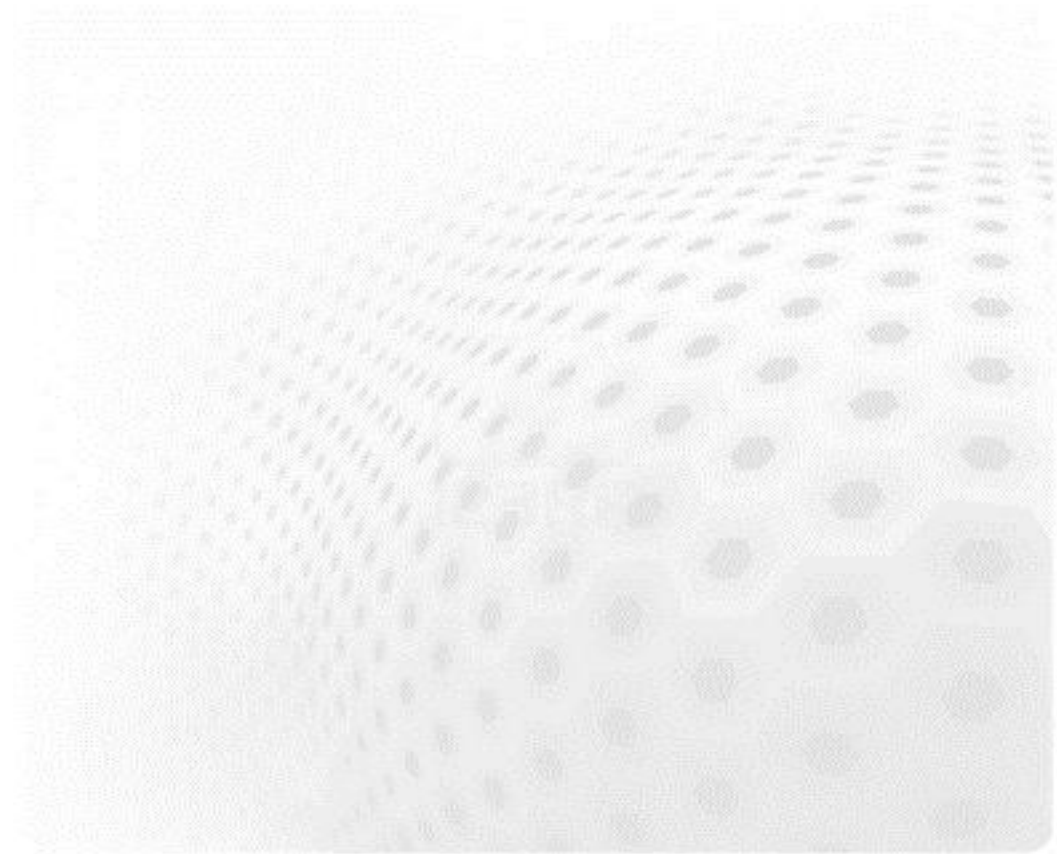


SCIPY



Scipy – Scientific tools for Python

- Scipy is a Python package containing several tools for scientific computing
- Modules for:
 - statistics, optimization, integration, interpolation
 - linear algebra, Fourier transforms, signal and image processing
 - ODE solvers, special functions
 - ...
- Vast package, reference guide is currently 975 pages
- Scipy is built on top of Numpy

Library overview

- Clustering package (`scipy.cluster`)
- Constants (`scipy.constants`)
- Fourier transforms (`scipy.fftpack`)
- Integration and ODEs (`scipy.integrate`)
- Interpolation (`scipy.interpolate`)
- Input and output (`scipy.io`)
- Linear algebra (`scipy.linalg`)
- Maximum entropy models (`scipy.maxentropy`)
- Miscellaneous routines (`scipy.misc`)
- Multi-dimensional image processing (`scipy.ndimage`)
- Orthogonal distance regression (`scipy.odr`)
- Optimization and root finding (`scipy.optimize`)
- Signal processing (`scipy.signal`)
- Sparse matrices (`scipy.sparse`)
- Sparse linear algebra (`scipy.sparse.linalg`)
- Spatial algorithms and data structures (`scipy.spatial`)
- Special functions (`scipy.special`)
- Statistical functions (`scipy.stats`)
- Image Array Manipulation and Convolution (`scipy.stsci`)
- C/C++ integration (`scipy.weave`)

Integration

- Routines for numerical integration
 - single, double and triple integrals
- Function to integrate can be given by function object or by fixed samples

integrate.py

```
from scipy.integrate import simps, quad, inf

x = np.linspace(0, 1, 20)
y = np.exp(-x)
int1 = simps(y, x)      # integrate function given by samples

def f(x):
    return exp(-x)

int2 = quad(f, 0, 1)    # integrate function object
int3 = quad(f, 0, inf) # integrate up to infinity
```

Optimization

- Several classical optimization algorithms
 - Quasi-Newton type optimizations
 - Least squares fitting
 - Simulated annealing
 - General purpose root finding
 - ...

```
>>> from scipy.optimize import fmin
>>>
```

Special functions

- Scipy contains huge set of special functions
 - Bessel functions
 - Legendre functions
 - Gamma functions
 - ...

```
>>> from scipy.special import jv, gamma
>>>
```

Linear algebra

- Wider set of linear algebra operations than in Numpy
 - decompositions, matrix exponentials
- Routines also for sparse matrices
 - storage formats
 - iterative algorithms

sparse.py

```
import numpy as np
from scipy.sparse.linalg import LinearOperator, cg

# "Sparse" matrix-vector product
def mv(v):
    return np.array([ 2*v[0], 3*v[1]])

A = LinearOperator( (2,2), matvec=mv, dtype=float )
b = np.array((4.0, 1.0))
x = cg(A, b)      # Solve linear equation Ax = b with conjugate gradient
```

Summary

- Scipy is vast package of tools for scientific computing
- Uses lots of NumPy in the background
- Numerical integration, optimization, special functions, linear algebra, ...
- Look Scipy documentation for finding tools for your needs!