

SciPy & other packages

Arno Proeme, ARCHER CSE Team

aproeme@epcc.ed.ac.uk

Attributed to Jussi Enkovaara &

Martti Louhivuori, CSC Helsinki



Reusing this material



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

http://creativecommons.org/licenses/by-nc-sa/4.0/deed.en_US

This means you are free to copy and redistribute the material and adapt and build on the material under the following terms: You must give appropriate credit, provide a link to the license and indicate if changes were made. If you adapt or build on the material you must distribute your work under the same license as the original.

Note that this presentation contains images owned by others. Please seek their permission before reusing these images.



EPSRC

NERC SCIENCE OF THE ENVIRONMENT

CRAY
THE SUPERCOMPUTER COMPANY

| epcc |



<http://www.archer.ac.uk>
support@archer.ac.uk



SciPy

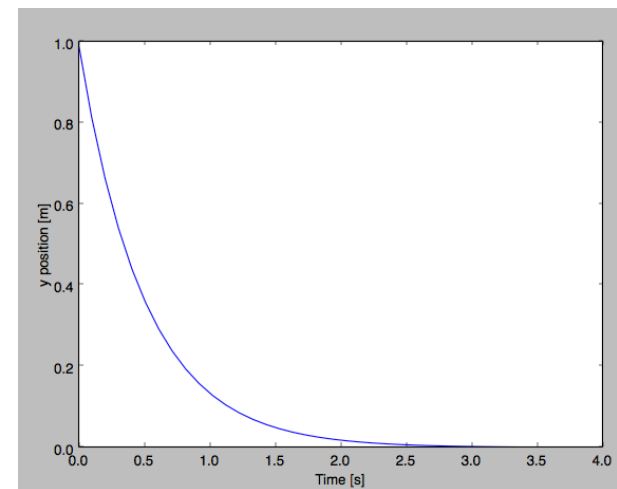
- NumPy provides arrays, basic linear algebra, random number generation, and Fourier transforms
- SciPy builds on NumPy (e.g. by using arrays) and expands this with (additional) routines for:
 - Numerical integration
 - Interpolation
 - Linear algebra and wrappers to LAPACK & BLAS
 - Sparse linear algebra
 - Image processing
 - Optimisation
 - Signal processing
 - Statistical functions
 - Spatial data structures and algorithms
 - Airy functions
- Note: no PDE solvers (though other packages exist)



Integration

- Routines for numerical integration – single, double and triple integrals
- Function to integrate can be given by function object or by fixed samples
- e.g. solve the ODE
 - $dy/dt = -2y$ between $t = 0..4$, with the initial condition $y(t=0) = 1$

```
import numpy as np
from scipy.integrate import odeint
def calc_derivative(ypos, time):
    return -2*ypos
time_vec = np.linspace(0, 4, 40)
yvec = odeint(calc_derivative, 1, time_vec)
pl.plot(time_vec, yvec)
```



Optimisation

- Several classical optimisation algorithms
 - Quasi-Newton type optimisations
 - Least squares fitting
 - Simulated annealing
 - General purpose root finding
- Rosenbrock function

$$f(\mathbf{x}) = \sum_{i=1}^{N-1} (x_i - x_{i-1}^2)^2 + (1 - x_{i-1})^2$$

```
>>> from scipy.optimize import fmin
>>> def rosen(x):
...     return sum(100.0*(x[1:]-x[:-1]**2.0)**2.0 + (1-x[:-1])**2.0)
>>> x0 = [1.3, 0.7, 0.8, 1.9, 1.2]
>>> xopt = fmin(rosen, x0, xtol=1e-8)
```

Optimization terminated successfully.

Current function value: 0.000000

Iterations: 339

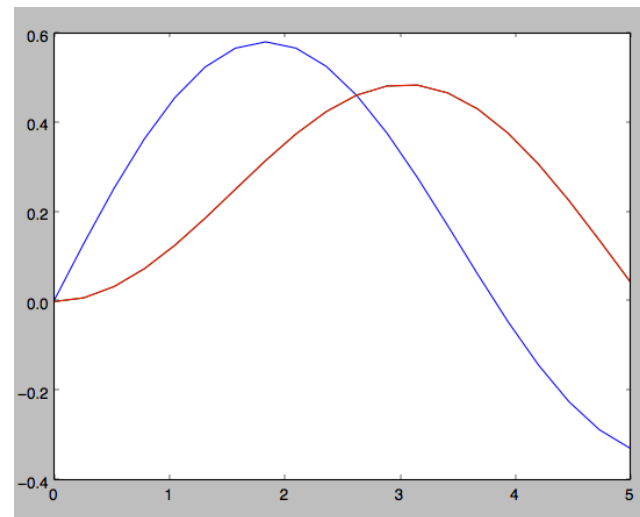
Function evaluations: 571



Special functions

- SciPy contains huge set of special functions – Bessel functions
 - Legendre functions
 - Gamma functions
 - Bessel function →

```
>>> from scipy.special import *  
>>> x = np.linspace(0, 5, 20)  
>>> plot(x, jv(1, x))  
>>> plot(x, jv(2, x))
```



Linear Algebra

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

```
>>> import numpy as np
>>> from scipy.sparse.linalg import LinearOperator, cg
>>> # Define "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
>>> x
(array([ 2.          ,  0.33333333]), 0)
```



Other packages

- Pandas
 - Offers R-like statistical analysis of numerical tables and time series
- SymPy
 - Python library for symbolic computing
- scikit-image
 - Advanced image processing
- scikit-learn
 - Package for machine learning
- Sage
 - Open source replacement for Mathematica / Maple / Matlab (built using Python)

