

**PHYS4038/MLiS and ASI/MPAGS**

# Scientific Programming in



[mpags-python.github.io](https://mpags-python.github.io)

**Steven Bamford**



**University of  
Nottingham**  
UK | CHINA | MALAYSIA

An introduction to scientific programming with



**Session 5:**  
Scientific Python

# Scientific Python (SciPy)

- Suite of numerical and scientific tools for Python
- <http://scipy.org/>
- <http://docs.scipy.org/>

The screenshot shows the SciPy.org website homepage. At the top, there is a blue header with the SciPy logo, the text "SciPy.org", and a "Sponsored By ENTHOUGHT" logo. A search bar and "Titles" and "Text" buttons are also present. Below the header, there is a navigation menu on the left with links to "Wiki", "Documentation", "Mailing Lists", "Download", "Installing SciPy", "Topical Software", "Cookbook", "Developer Zone", "Blogs", "Conference", and "SciPy". The main content area features a "SciPy" heading and five icons with labels: "Download", "Getting Started", "Documentation", "Report Bugs", and "Read the Blog". Below this is a section titled "Scientific Tools for Python" with a paragraph of text. On the right side, there is a "News" section with several entries about NumPy releases and a conference call for papers.

SciPy.org Search  Titles Text

Wiki

- Documentation
- Mailing Lists
- Download
- Installing SciPy
- Topical Software
- Cookbook
- Developer Zone
- Blogs
- Conference
- SciPy

Page

- Immutable Page
- Info
- Attachments
- More Actions:

## SciPy

[Download](#) [Getting Started](#) [Documentation](#) [Report Bugs](#) [Read the Blog](#)

### Scientific Tools for Python

SciPy (pronounced "Sigh Pie") is open-source software for mathematics, science, and engineering. It is also the name of a very popular [conference](#) on scientific programming with Python. The SciPy library depends on [NumPy](#), which provides convenient and fast N-dimensional array manipulation. The SciPy library is built to work with NumPy arrays, and provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization. Together, they run on all popular operating systems, are quick to install, and are free of charge. NumPy and SciPy are easy to use, but powerful enough to be depended upon by some of the world's leading scientists and engineers. If you need to manipulate numbers on a computer and display or publish the results, give SciPy a try!

#### News

- Python for Scientific Computing Conference 2009: Call for papers.**
- NumPy 1.3.0 released.** (2009-04-05) See the [Download](#) and [Release Notes](#) pages.
- NumPy 1.3.0 rc2 released.** (2009-04-03) See the [Download](#) and [Release Notes](#) pages.
- NumPy 1.3.0 rc1 released.** (2009-03-28) See the [Download](#) and [Release Notes](#) pages.
- NumPy 1.3.0 beta 1 released.** (2009-03-19) See the [Download](#) and [Release Notes](#) pages.
- SciPy 0.7.0 released.** (2009-02-11) See the [Download](#)

# Scipy subpackages

- cluster Clustering algorithms
- constants Physical and mathematical constants
- fftpack Fast Fourier Transform routines
- integrate Integration and ordinary differential equation solvers
- interpolate Interpolation and smoothing splines
- io Input and Output
- linalg Linear algebra
- ndimage N-dimensional image processing
- odr Orthogonal distance regression
- optimize Optimization and root-finding
- signal Signal processing
- sparse Sparse matrices and associated routine
- spatial Spatial data structures and algorithms
- special Special functions
- stats Statistical distributions and functions

```
# scipy submodules  
# must be explicitly  
# imported, e.g.,  
import scipy.fftpack  
# or  
from scipy import stats
```

Some simple examples:

- Special functions (special)
- Root finding (optimize)
- Integration (integrate)
- Statistics (stats)
- Image processing (ndimage)
- Interpolation (interpolate)
- Optimisation (optimize)

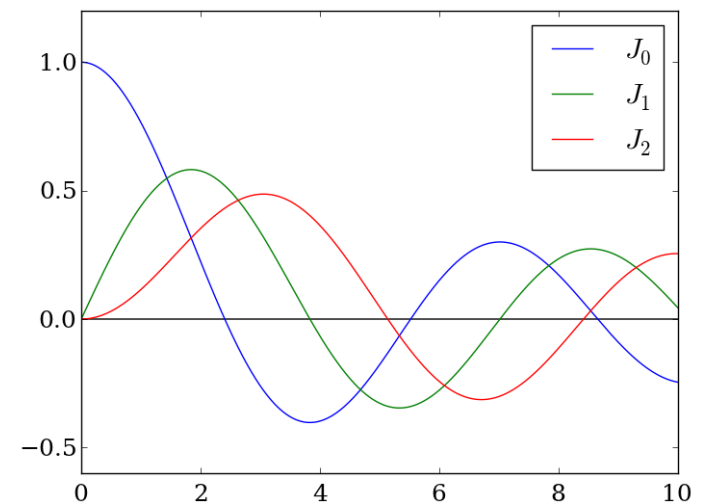
# Scipy – special functions

- Huge number of functions, including...
  - Bessel functions
  - Gamma functions
  - Fresnel integrals
  - Hypergeometric functions
  - Orthogonal polynomials

*e.g., Bessel functions of order 1, 2, 3*

```
>>> from scipy import special
>>> x = np.arange(0, 10.001, 0.01)
>>> for alpha in range(3):
...     y = special.jv(alpha, x)
...     plt.plot(x, y,
...              label=r'$J_{%i}$'%alpha)
>>> plt.hlines(0, 0, 10)
>>> plt.legend()
```

$$x^2 \frac{d^2 y}{dx^2} + x \frac{dy}{dx} + (x^2 - \alpha^2)y = 0$$



# Scipy – root finding

- Accurate automatic root-finding using MINPACK

```
>>> from scipy.optimize import fsolve # n-dimensional root finder
>>> from scipy.special import jv
```

*Define a function to solve*

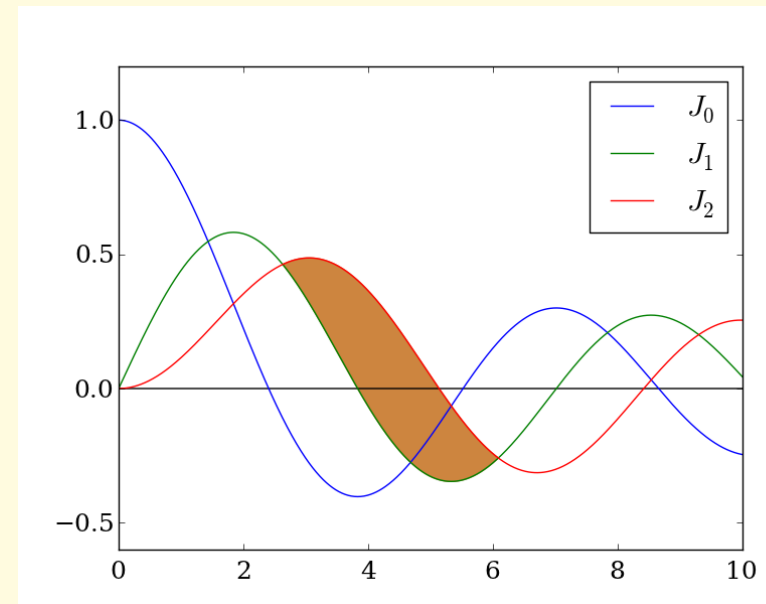
*First argument is variable (or array of variables) of interest*

```
>>> def f(z, a1, a2):
...     return jv(a1, z) - jv(a2, z)
...
```

```
>>> fsolve(f, 2.5, args=(1, 2))
array([ 2.62987411])
```

```
>>> fsolve(f, 6, args=(1, 2))
array([ 6.08635978])
```

```
>>> plt.fill_between(x, special.jv(1, x), special.jv(2, x),
                    where=((x > 2.630) & (x < 6.086)), color="peru")
```



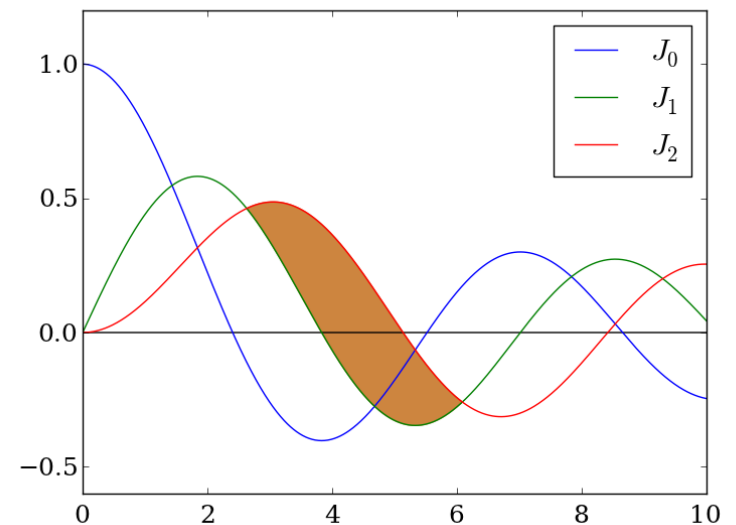
# Scipy – integration

- Accurate automatic integration using QUADPACK
  - including uncertainty estimate

```
>>> from scipy.integrate import quad # one-dimensional integration
Using previous function (first argument is variable of interest)
>>> r = fsolve(f, (2.5, 6), args=(1, 2))

>>> print r
[ 2.62987411  6.08635978]

>>> quad(f, r[0], r[1], args=(1, 2))
(-0.98961158607157, 1.09868956829247e-14)
```



- Can specify limits at infinity  
(-np.inf, np.inf)

```
>>> quad(exp, -np.inf, 0)
(1.000000000000000002, 5.842606742906004e-11)
```



# Scipy – integration

- QUADPACK and MINPACK routines provide warning messages
- Extra details returned if parameter `full_output=True`

```
>>> quad(tan, 0, pi/2.0-0.0001)
(9.210340373641296, 2.051912874185855e-09)
```

```
>>> quad(tan, 0, pi/2.0)
Warning: Extremely bad integrand behavior occurs at some points of the
integration interval.
(38.58895946215512, 8.443496712555953)
```

```
>>> quad(tan, 0, pi/2.0+0.0001)
Warning: The maximum number of subdivisions (50) has been achieved.
If increasing the limit yields no improvement it is advised to analyze
the integrand in order to determine the difficulties. If the position of a
local difficulty can be determined (singularity, discontinuity) one will
probably gain from splitting up the interval and calling the integrator
on the subranges. Perhaps a special-purpose integrator should be used.
(6.896548923283743, 2.1725421039565056)
```

# Scipy – statistics

- Probability distributions
  - including: norm, chi2, t, expon, poisson, binom, boltzmann, ...
  - methods:
    - rvs – return array of random variates
    - pdf – probability density function
    - cdf – cumulative density function
    - ppf – percent point function
    - ... and many more
- Statistical functions
  - including:
    - mean, median, skew, kurtosis, ...
    - normaltest, probplot, ...
    - pearsonr, spearmanr, wilcoxon, ...
    - ttest\_1samp, ttest\_ind, ttest\_rel, ...
    - kstest, ks\_2samp, ...

```
>>> lambda = 10
>>> p = stats.poisson(lambda)

# P(n > 20)
>>> 1 - p.cdf(20)
0.0015882606618580573

# N: P(n < N) = 0.05, 0.95
>>> p.ppf((0.05, 0.95))
array([ 5., 15.])

# true 95% CI bounds on lambda
>>> stats.gamma.ppf((0.025, 0.975),
                    lambda+0.5, 1)
array([ 6.14144889, 18.73943795])
```

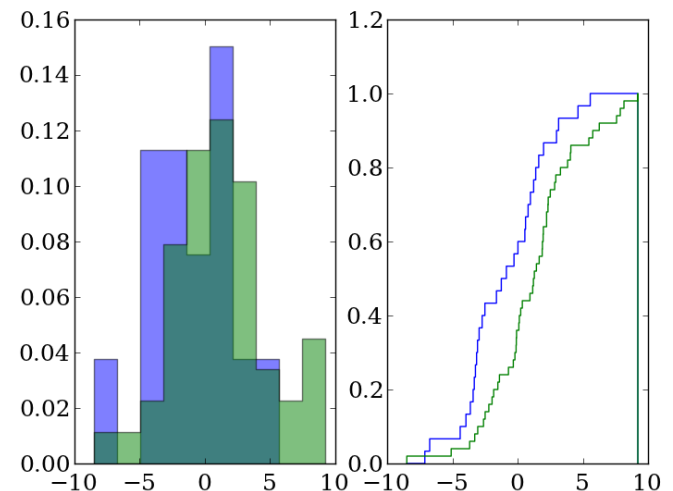
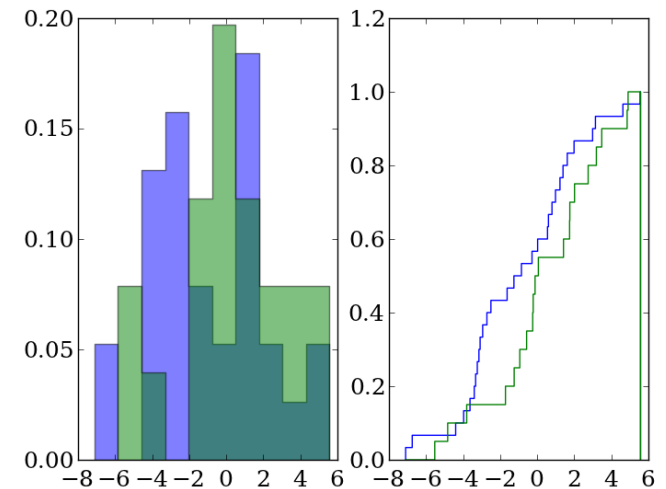
# Scipy – statistics

```
>>> x = stats.norm.rvs(-1, 3, size=30) # specify pdf parameters
>>> n = stats.norm(1, 3) # create 'frozen' pdf
>>> y = n.rvs(20)
>>> z = n.rvs(50)
>>> p = plt.subplot(121)
>>> h = plt.hist((x, y), normed=True,
                histtype='stepfilled', alpha=0.5)
>>> p = plt.subplot(122)
>>> h = plt.hist((x, y), histtype='step',
                cumulative=True, normed=True, bins=1000)

>>> stats.ks_2samp(x, y)
(0.29999999999999999, 0.18992875018013033)
>>> stats.ttest_ind(x, y)
(-1.4888787966012809, 0.14306062943339182)

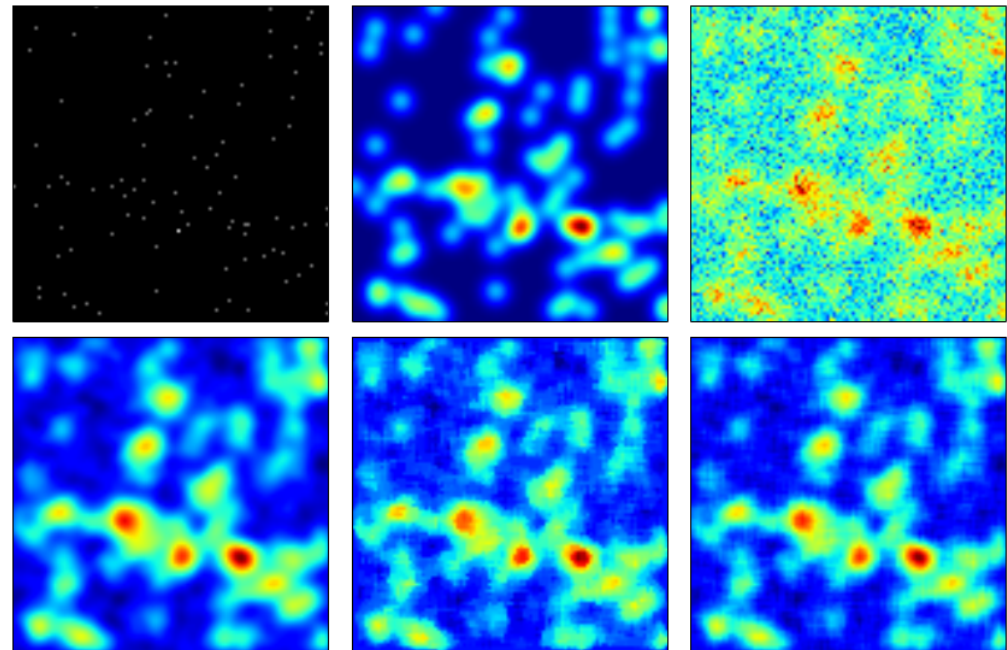
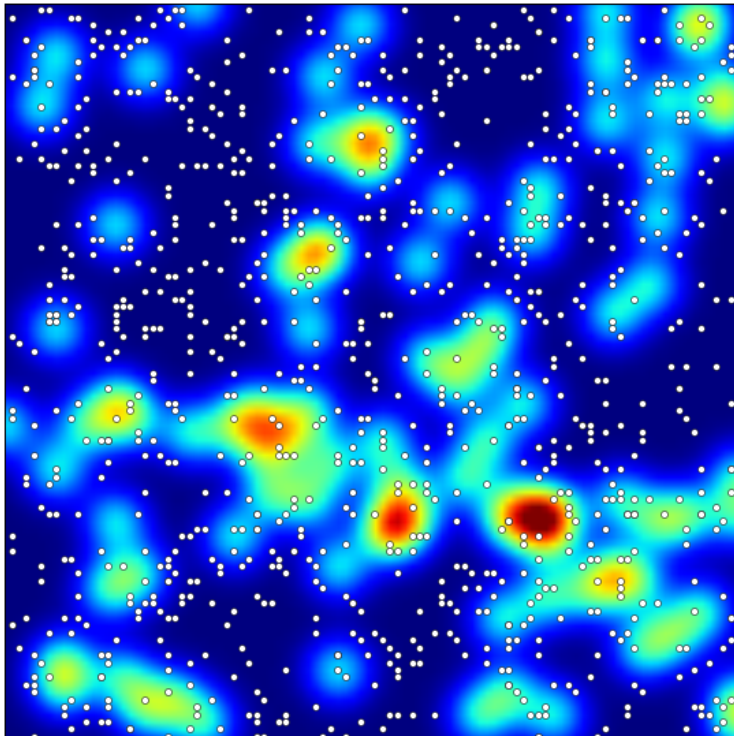
>>> stats.ks_2samp(x, z)
(0.31333333333333335, 0.039166429989206733)
>>> stats.ttest_ind(x, z)
(-2.7969511393118509, 0.0064942129302196124)

>>> stats.kstest(x, stats.norm(1, 3).cdf)
(0.3138899035681928, 0.0039905619713858087)
```



# Scipy – filtering and interpolation

## Notebook filtering and interpolation example [\[link to online notebook\]](#)



# Scipy – optimisation

- Local optimisation
  - `minimize` function
    - lots of options, different optimizers, constraints
- Least squares fitting
  - `curve_fit`
    - uses Levenberg-Marquardt algorithm

Details at <http://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html>

## Notebook fitting example

[\[link to online notebook\]](#)

**Other / more options...**

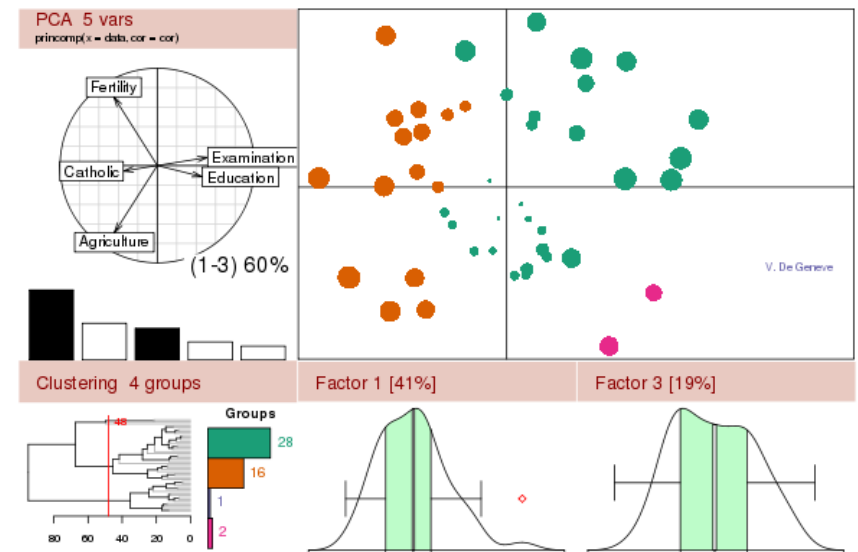


- Astronomical constants, units, times and dates
- Astronomical coordinate systems
- Cosmology calculations
- Virtual Observatory integration
- Astronomy specific additions to numpy/scipy tools:
  - n-dimensional datasets, tables
  - model fitting, convolution, filtering, statistics
- Open source, on GitHub

# RPy



- <http://rpy.sourceforge.net/>
- Wraps R – a statistics analysis language
  - very powerful
  - used by statisticians
  - many advanced stats capabilities
  - quite specialised
- <http://www.r-project.org>





# PyGSL

- Python wrappers of GNU Scientific Library functions
- PyGSL: <http://pygsl.sourceforge.net/>
- GSL: <http://www.gnu.org/software/gsl/>
  
- Incomplete documentation for Python functions, but almost all of GSL is wrapped, so refer to GSL documentation.
  
- Most functionality implemented in SciPy
  - or other, more Pythonic, tools
  - comprehensive and sometimes more tested

**PHYS4038/MLiS and ASI/MPAGS**

# Scientific Programming in



[mpags-python.github.io](https://mpags-python.github.io)

**Steven Bamford**



**University of  
Nottingham**  
UK | CHINA | MALAYSIA