```
>>> import numpy as np
>>> import scipy as sp

% ipython notebook --pylab inline
```

Scientific Programming I

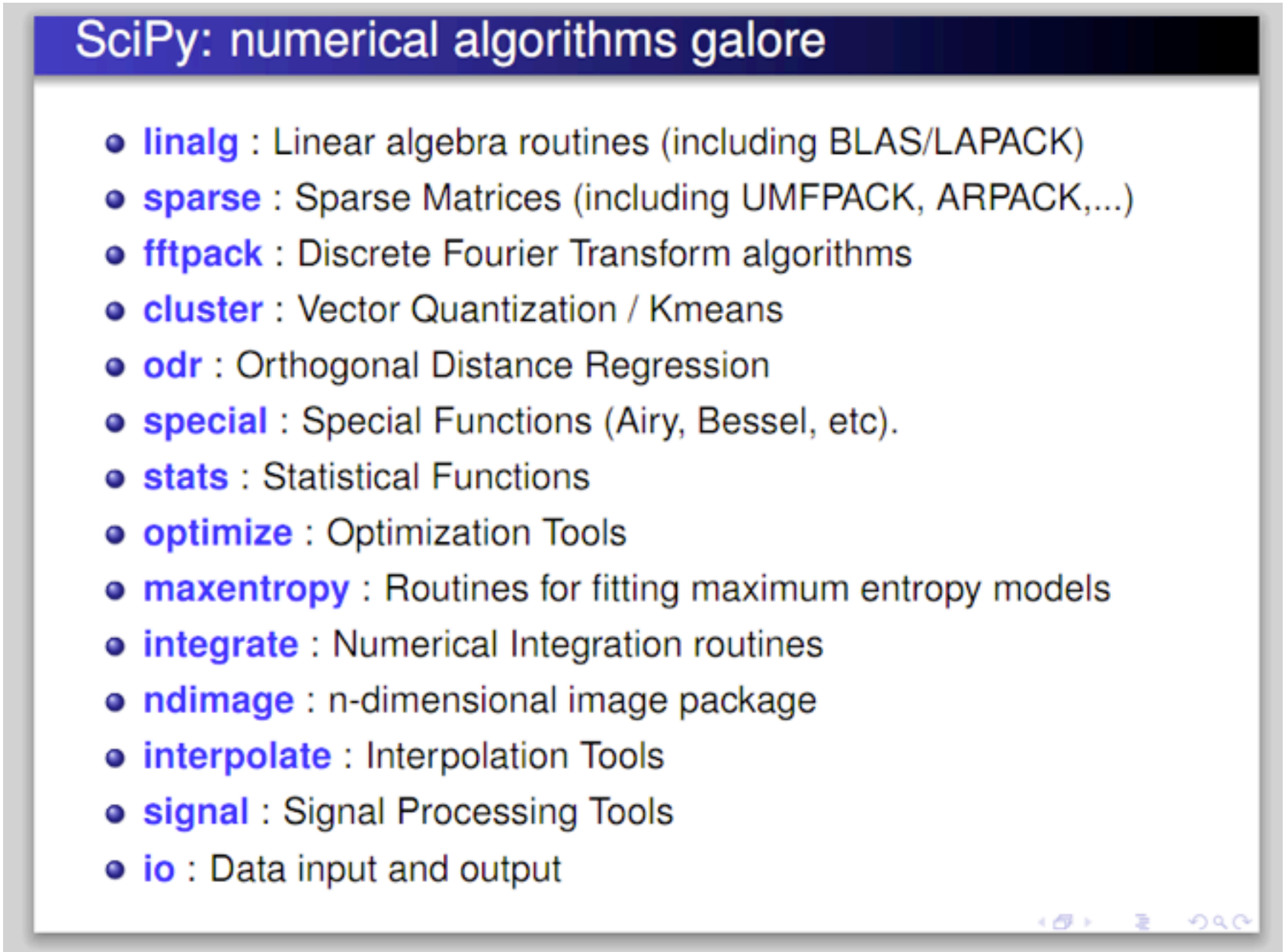
Berian James <berian@berekley.edu>

Astronomy Department, UC Berkeley

See also: Nat Butler’s lecture notes from last year’s bootcamp

SciPy packages

- This lecture explores:
  `linalg`, `fftpack`, `optimize`, `integrate` and `interpolate`



Overview of SciPy challenges in this lecture

- Generating random variables with the same distribution as an input data set
  Using: sp.integrate, sp.interpolate, np.random
  Presupposes: Some statistics background, but I will provide a primer

- Calculating a Fourier transform of 1D data, with error bars
  Using: sp.fftpack, np.random
  Presupposes: Knowledge of Fourier transformation

- Fitting a model to (perhaps covariant) data
  Using: sp.optimize, sp.linalg
  Presupposes: A bit more statistics background, which, again, I will describe

  scipy.*package_name*.[tab]
  scipy.*package_name*.*function_name*?
  http://docs.scipy.org/doc/scipy/reference/ *SciPy reference guide, tutorial*

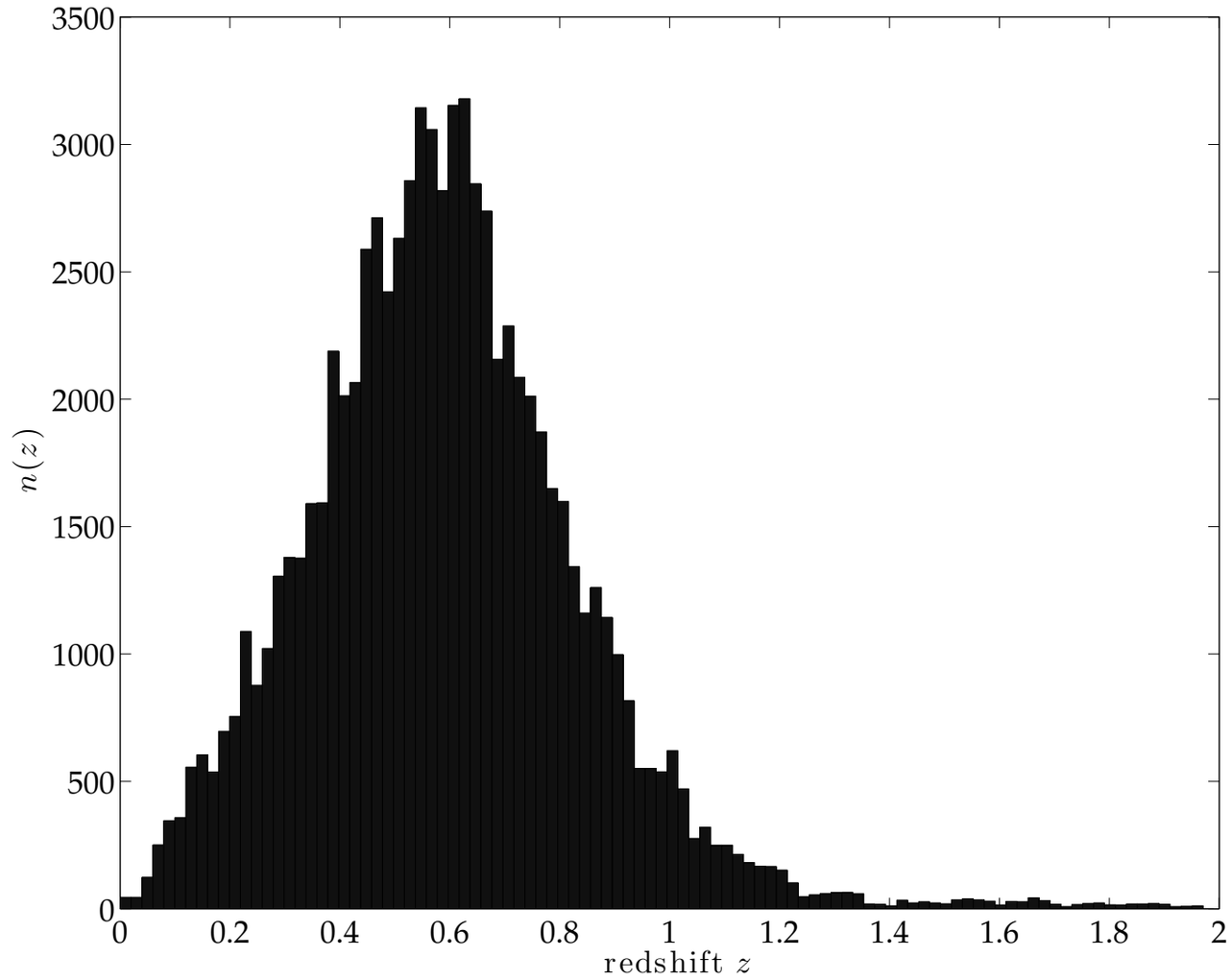
I. Integration and interpolation

```
>>> from scipy.integrate import *
>>> from scipy.interpolate import *
```

Problem: generating random variables from data

$$\text{Data: } [0.674, 1.053, 0.453 \ldots]$$

$$\text{PDF: } \hat{p}(z) \propto \texttt{hist}(\text{Data})$$

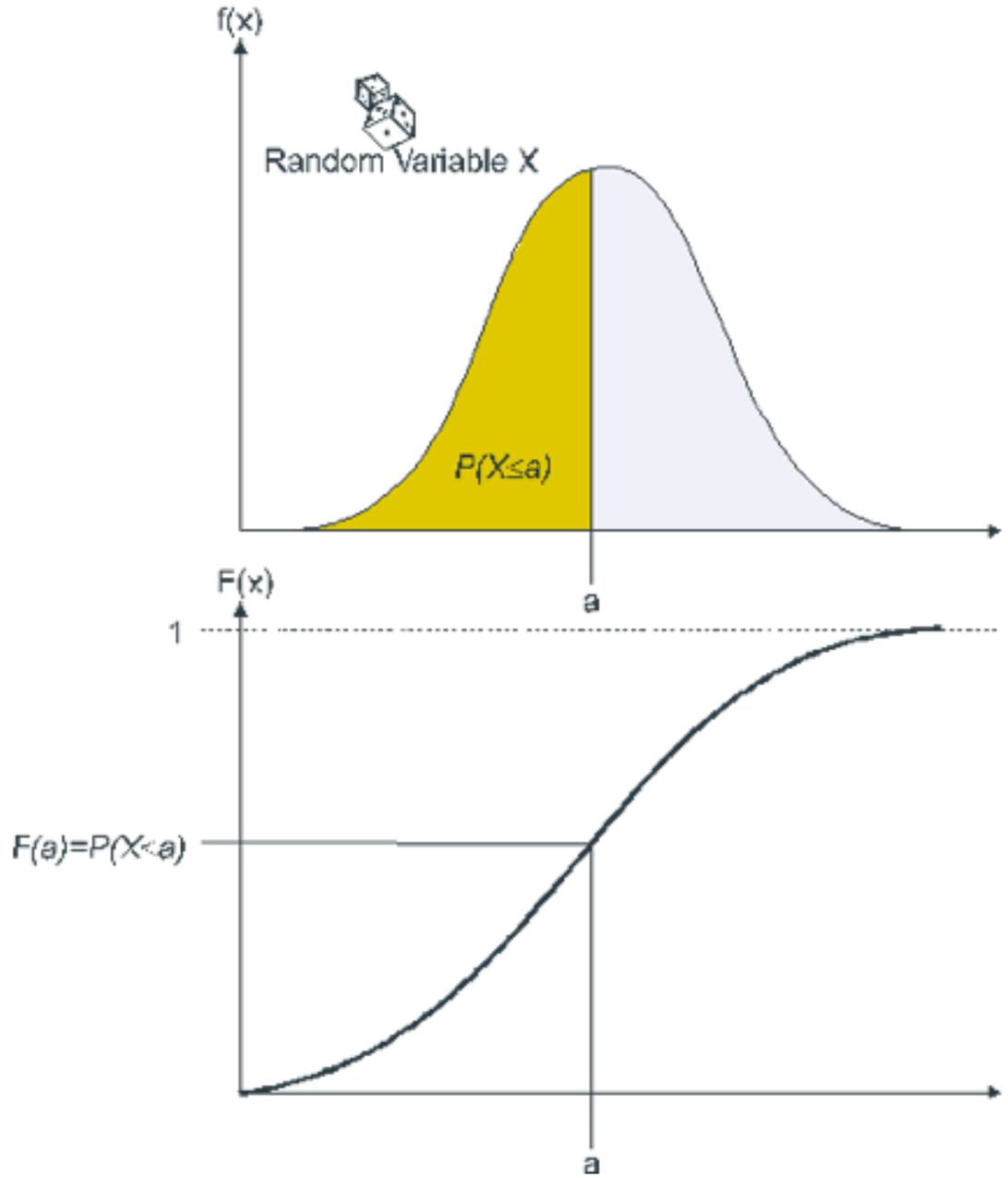


Problem: generating random variables from data

Data: $[0.674, 1.053, 0.453 \ldots]$

PDF: $\hat{p}(z) \propto \texttt{hist}(\text{Data})$

$$\text{CDF} : \hat{P}(z) = \int_0^z p(z')dz'$$

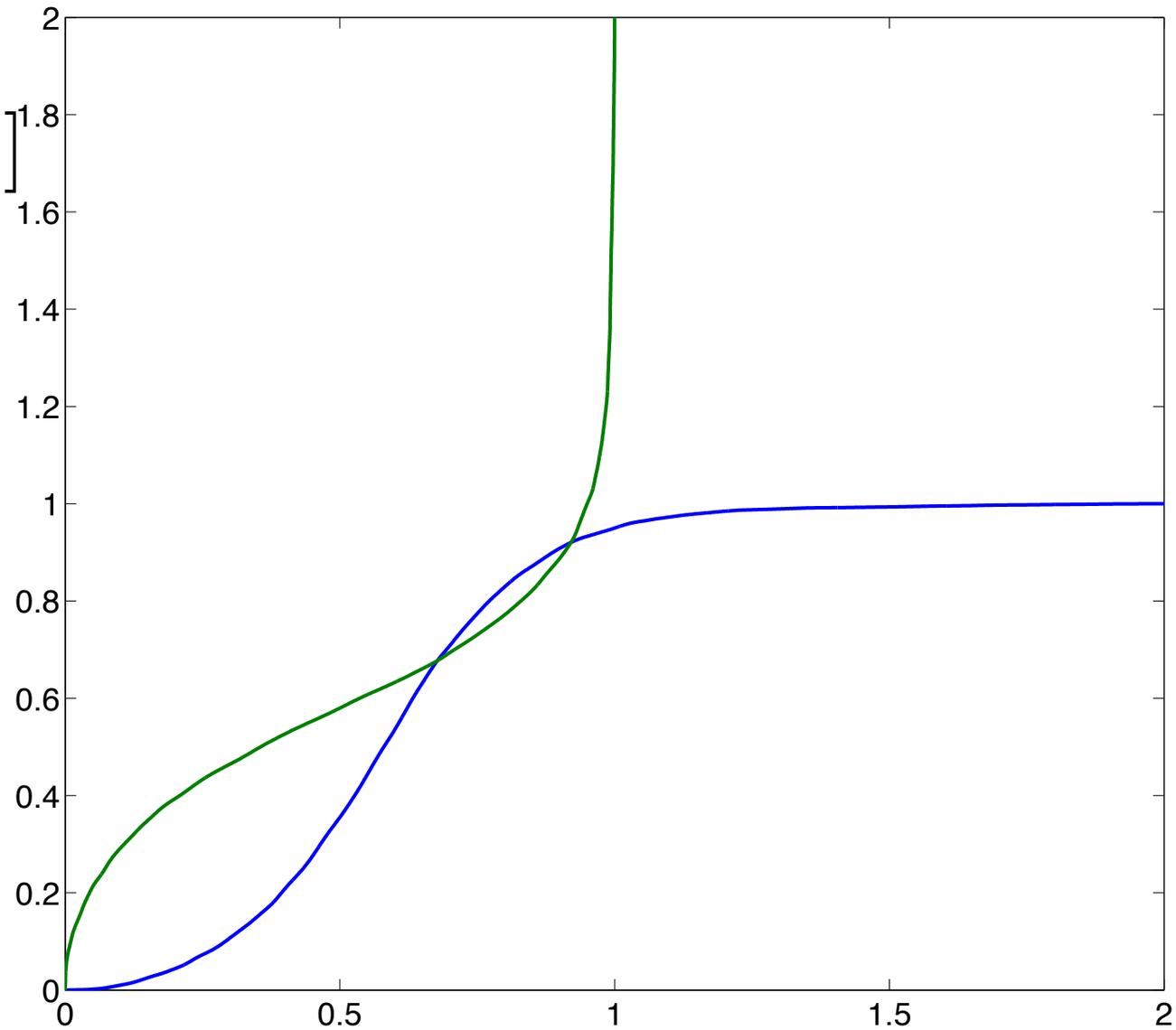


Problem: generating random variables from data

$$\text{Data: } [0.674, 1.053, 0.453 \ldots]$$

$$\text{PDF: } \hat{p}(z) \propto \texttt{hist}(\text{Data})$$

$$\text{CDF} : \hat{P}(z) = \int_0^z p(z')dz'$$

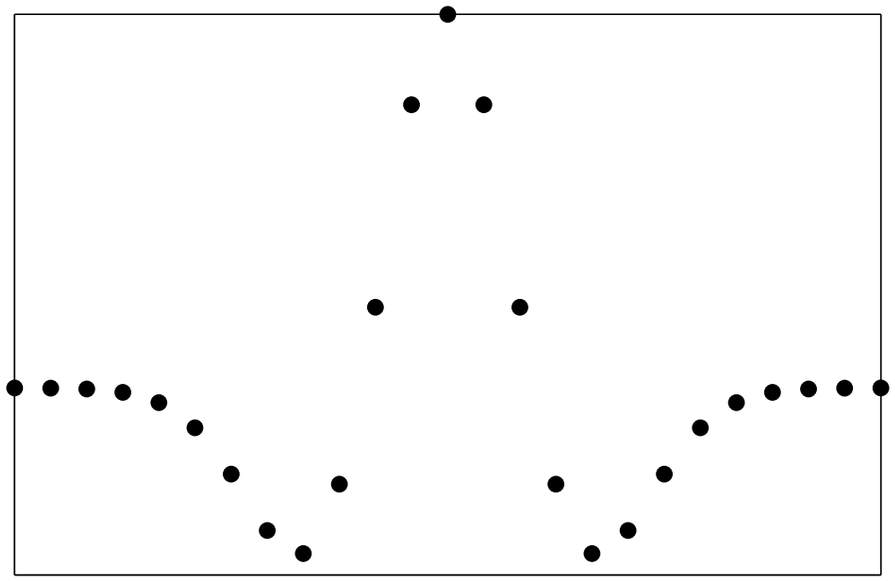


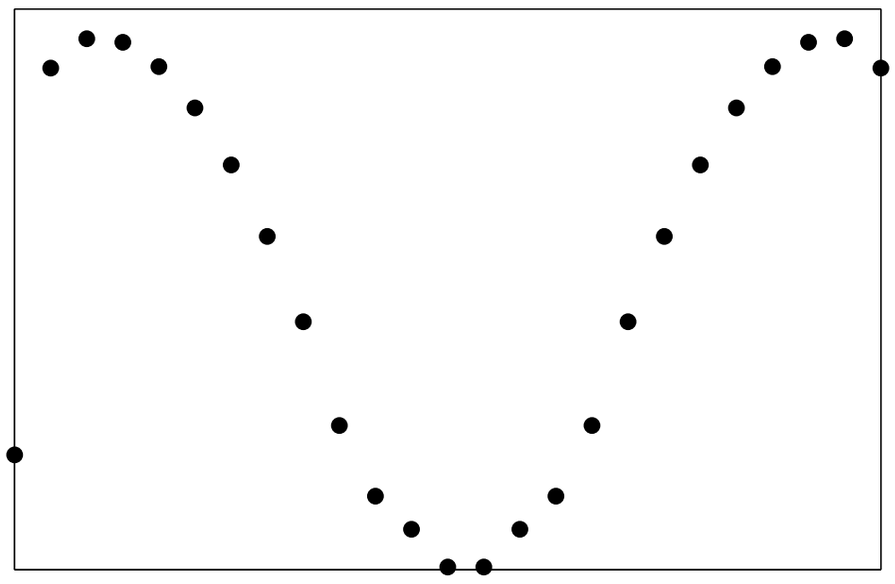
If $U \sim \text{Uniform}[0, 1)$, then $\hat{P}^{-1}(U) \sim \text{Data}$

II. An uncertain Fourier transform

```
>>> from numpy.random import *
>>> from scipy.fftpack import *
```

Problem: FFT with error bars

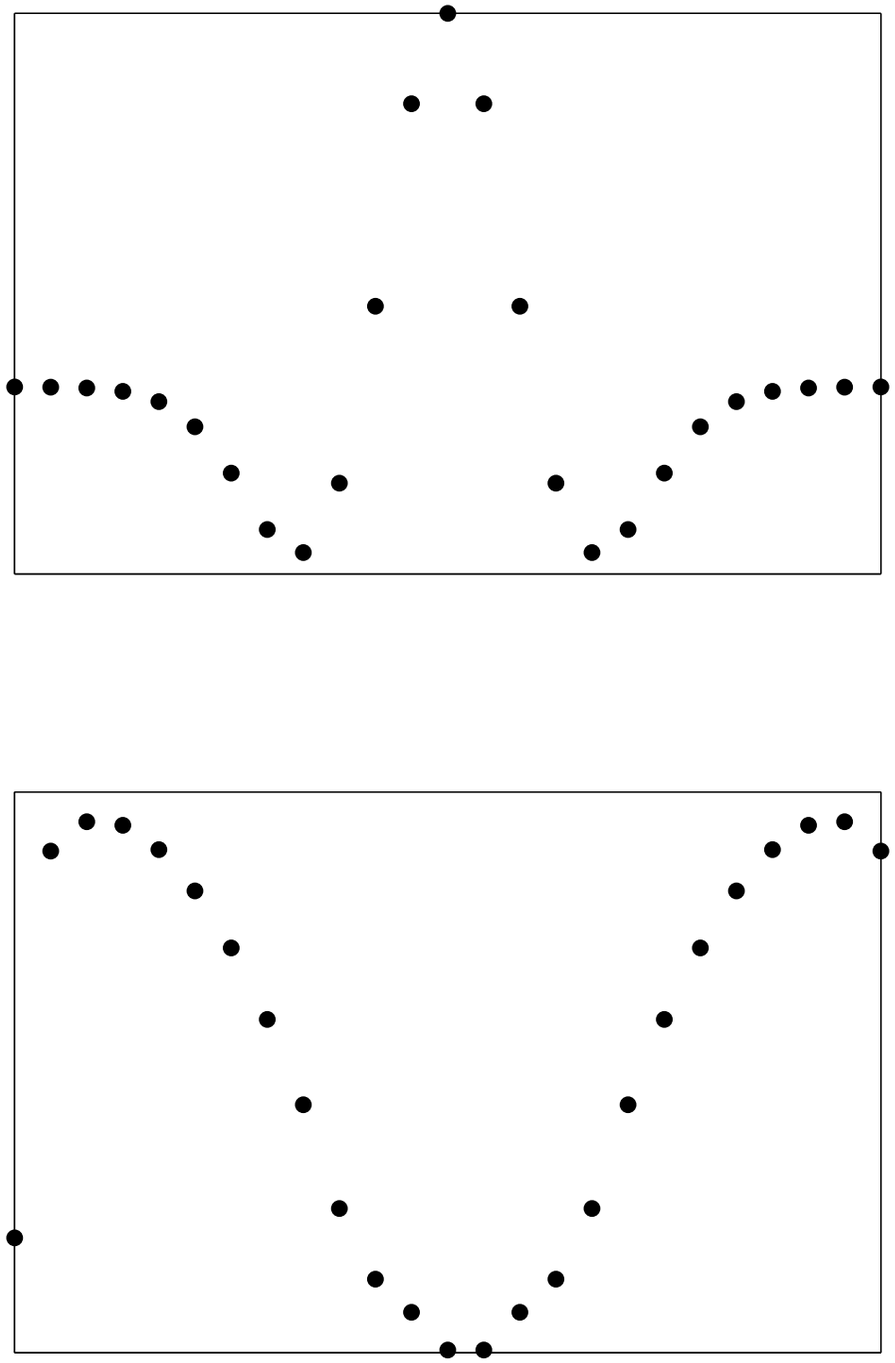


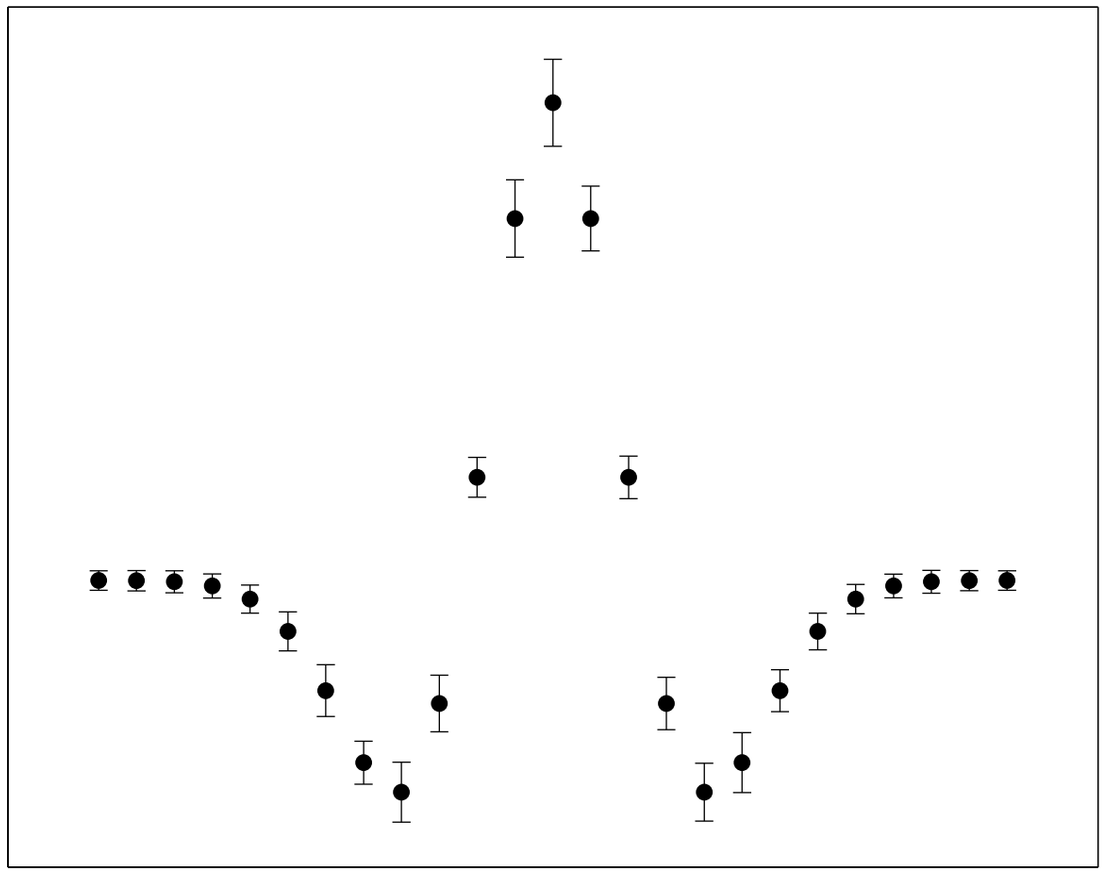


$$F(\tau) = \mathcal{F}(f(t))$$

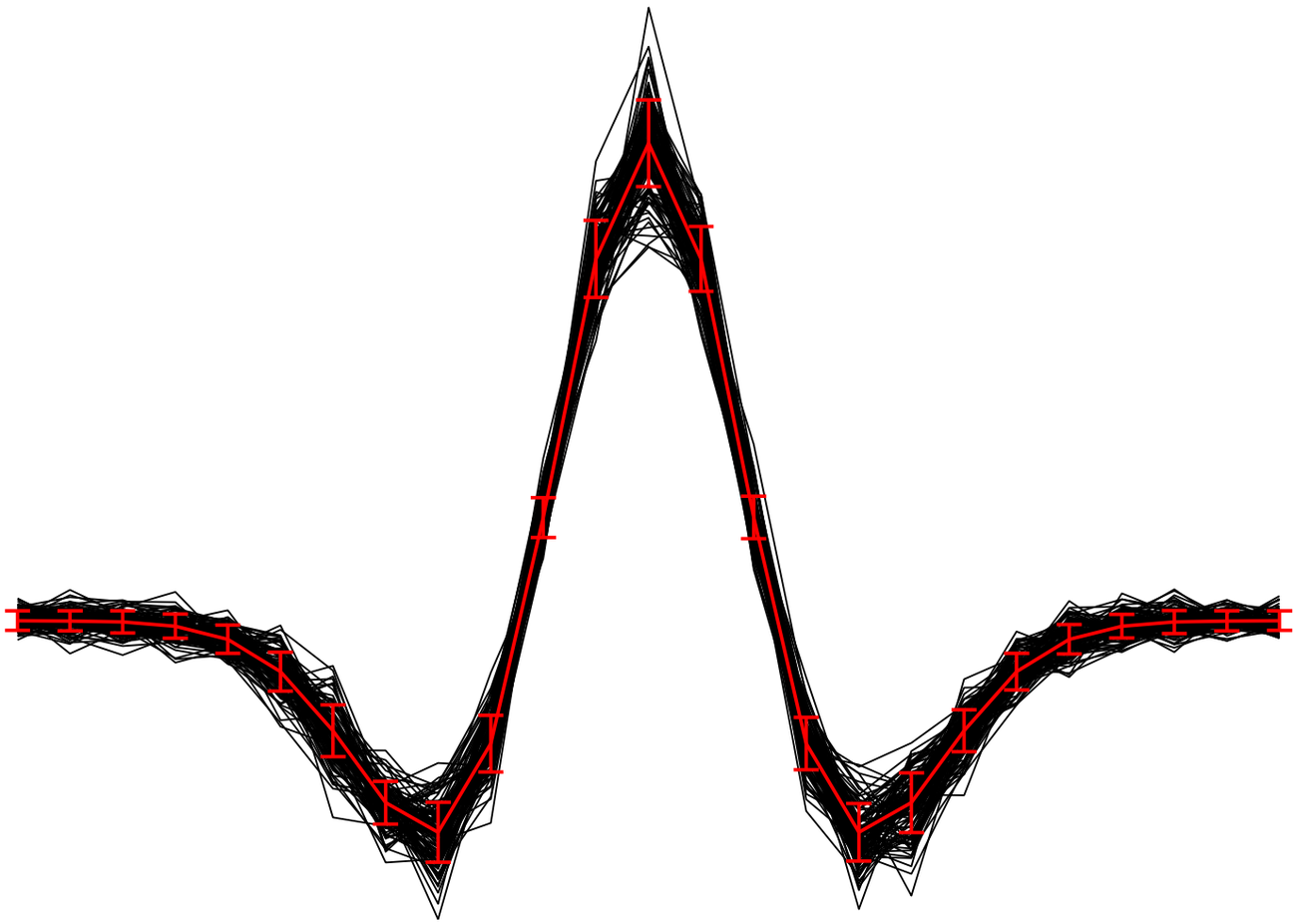
$$P(\tau) = F(\tau) \times F^*(\tau)$$

Problem: FFT with error bars





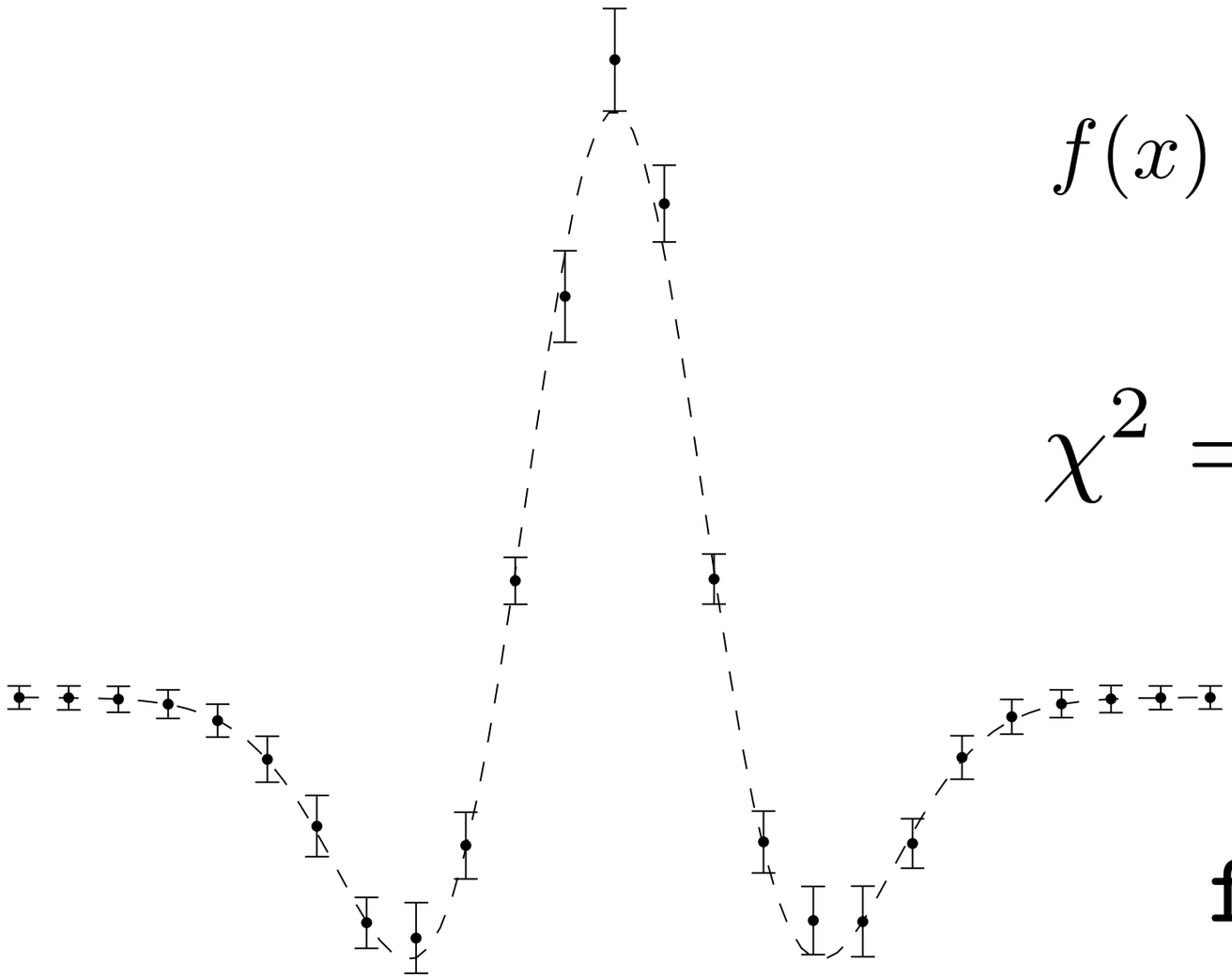
(Unsatisfactory) solution: brute sampling



III. Fitting (likelihood estimation) with covariant data

```
>>> import scipy.linalg as la
>>> import scipy.optimize as opt
```

Problem: non-linear model fitting

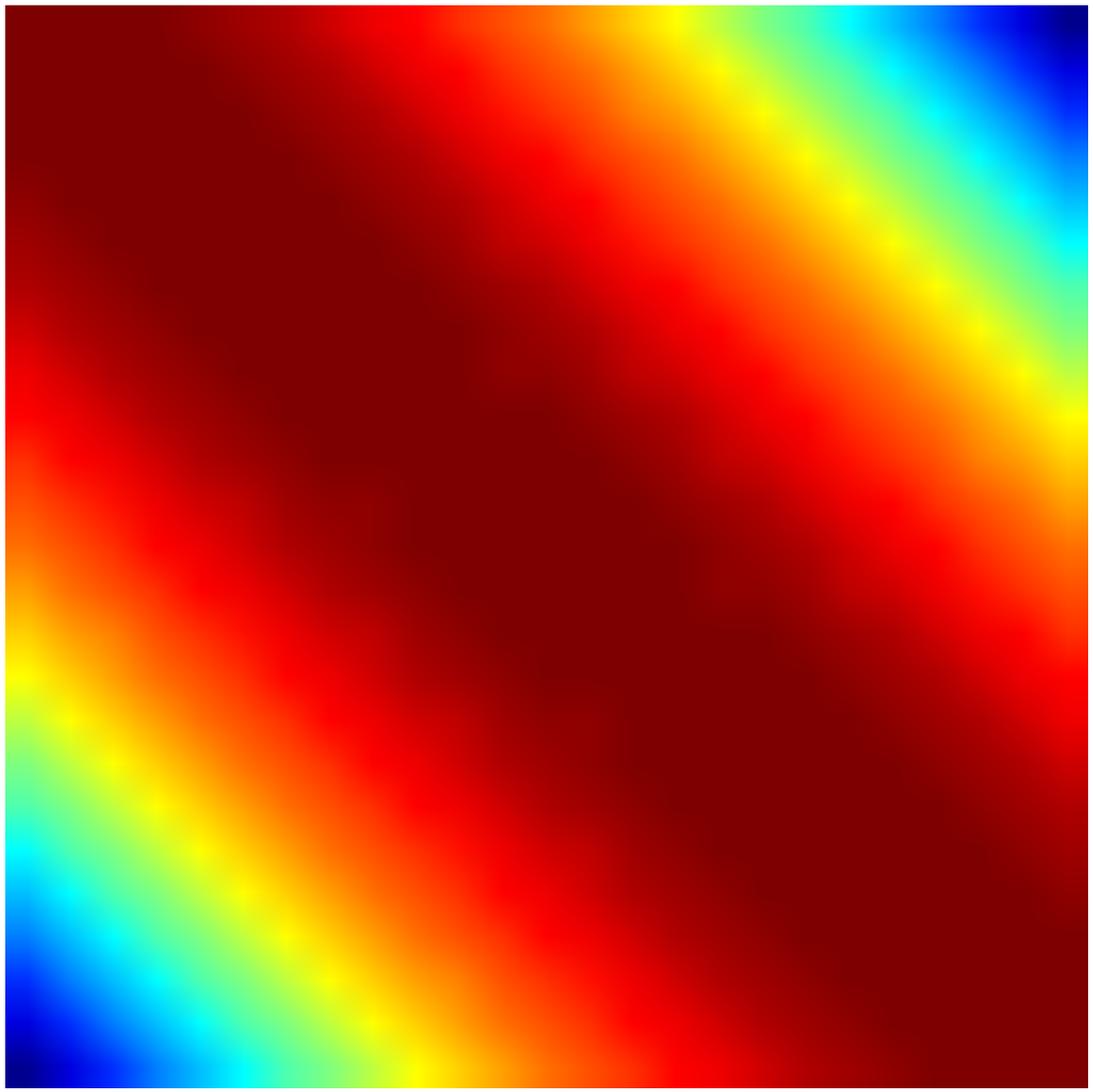


$$f(x) = P \exp\left(-\frac{x^2}{2}\right)(1-x^2)$$

$$\chi^2 = \sum_i \left(\frac{y_i - f(x_i)}{\sigma_i}\right)^2$$

```
fmin(chi2,[x0])
```

Problem: non-linear model fitting with covariance



$$f(x) = P \exp\left(-\frac{x^2}{2}\right)(1 - x^2)$$

$$\Delta = y_i - f(x_i)$$

$$\chi^2 = \Delta \mathbf{C}^{-1} \Delta'$$

```
fmin(chi2,[x0])
```

Eigendecomposition (for covariant data points)

- For a symmetric matrix, eigendecomposition can be used to deal with tricky inversions

- One method is to locate very small eigenvalues, and set their inverse to zero.

$$C = VEV' \Rightarrow C^{-1} = VE^{-1}V'$$