6. Advanced Numerical Methods

Part 1: Monte Carlo Methods

Part 2: Fourier Methods

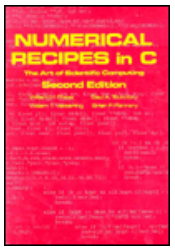
Part 1: Monte Carlo Methods

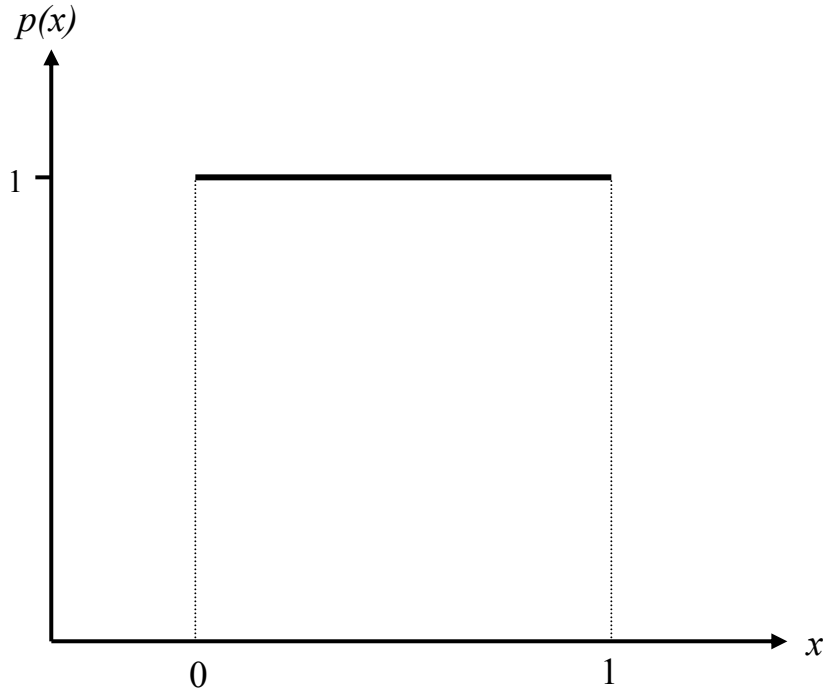
1. Uniform random numbers

Generating uniform random numbers, drawn from the pdf U[0,1], is fairly easy. Any scientific Calculator will have a RAN function…

Better examples of U[0,1] random number generators can be found in Numerical Recipes.

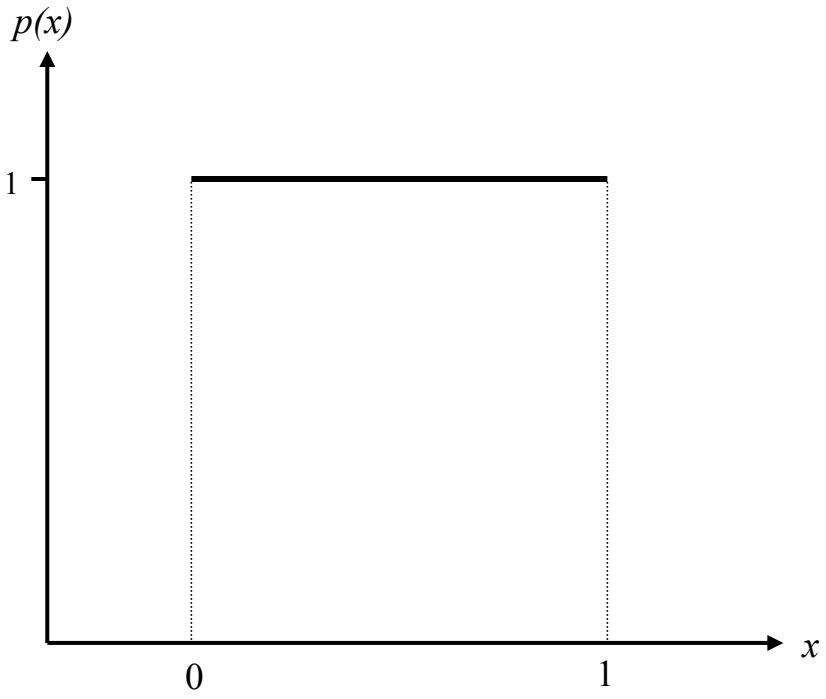
http://www.numerical-recipes.com/



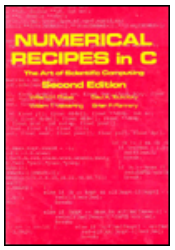


6.1. Uniform random numbers

Generating uniform random numbers, drawn from the pdf U[0,1], is fairly easy. Any scientific Calculator will have a RAN function…



Better examples of U[0,1] random number generators can be found in Numerical Recipes.



http://www.numerical-recipes.com/

In what sense are they better?…

Algorithms only generate **pseudo-random** numbers: very long (deterministic) sequences of numbers which are approximately random (i.e. no discernible pattern).

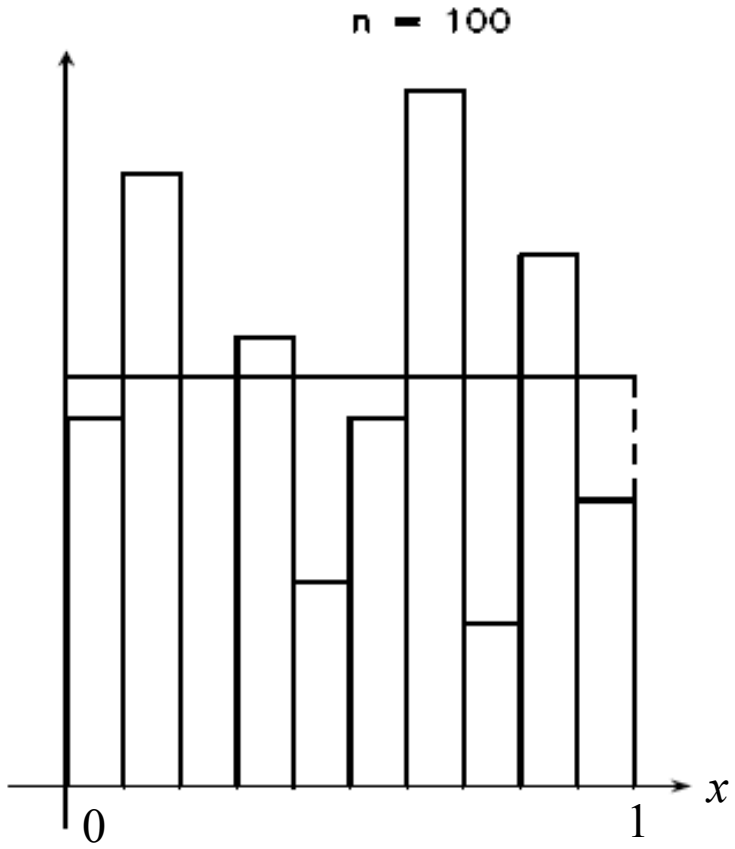
The better the RNG, the better it approximates U[0,1]

We can test pseudo-random numbers for randomness in several ways:

(a) Histogram of sampled values.

We can use hypothesis tests to see if the sample is consistent with the pdf we are trying to model.

e.g. Chi-squared test, applied to the
to the numbers in each histogram bin.



We can test pseudo-random numbers for randomness in several ways:

(a) Histogram of sampled values.

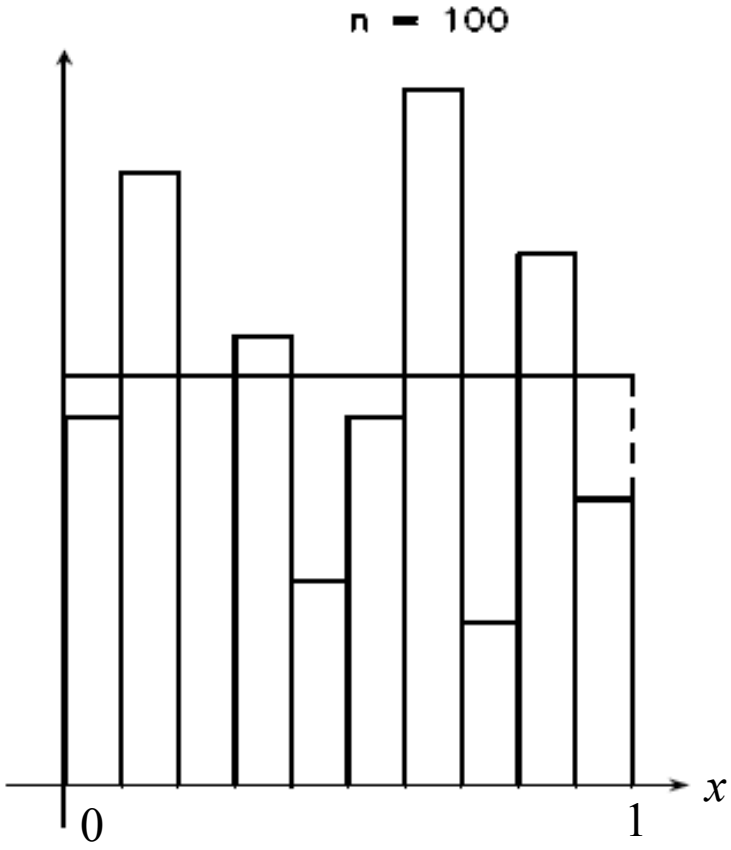
We can use hypothesis tests to see if the sample is consistent with the pdf we are trying to model.

e.g. Chi-squared test, applied to the to the numbers in each histogram bin.

$$\chi^2 = \sum_{i=1}^{n_{\text{bin}}} \left(\frac{n_i^{\text{obs}} - n_i^{\text{pred}}}{\sigma_i} \right)^2$$

Assume the bin number counts are subject to Poisson fluctuations, so that $\sigma_i^2 = n_i^{\text{pred}}$

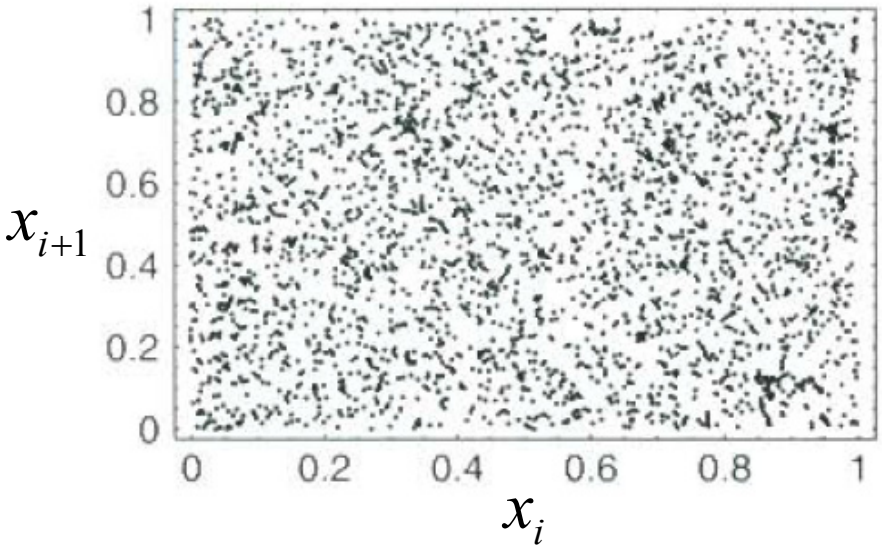
Note: no. of degrees of freedom $= n_{\text{bin}} - 1$ since we know the total sample size.

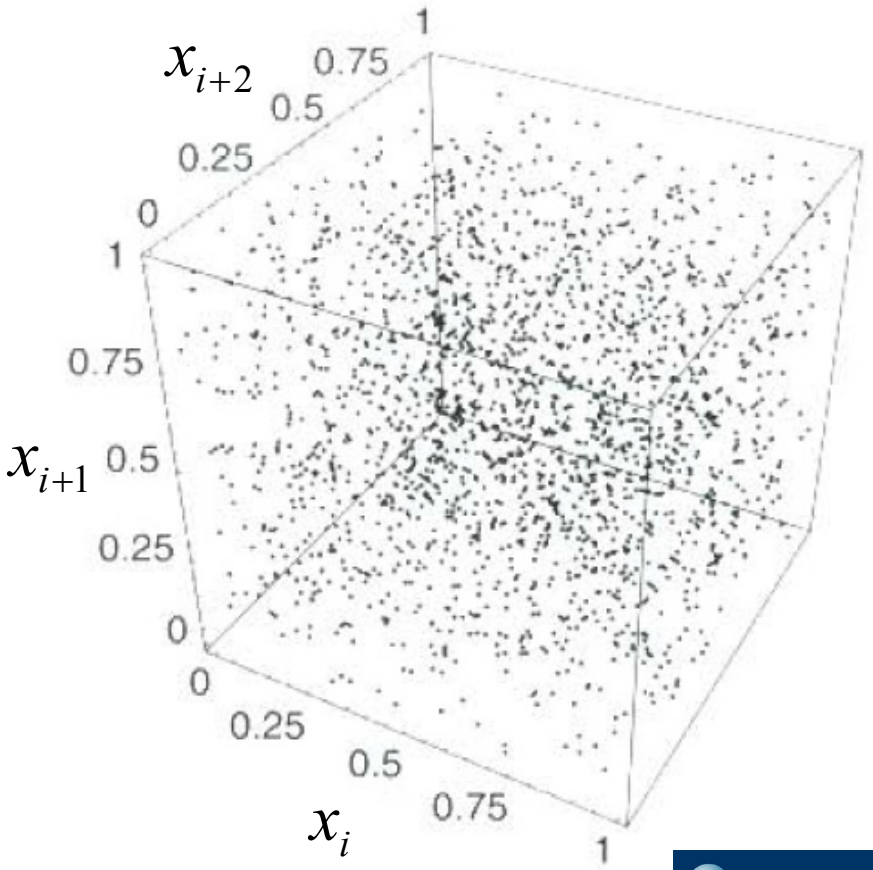


(b) Correlations between neighbouring pseudo-random numbers

Sequential patterns in the sampled values would show up as structure in the phase portraits – scatterplots of the i^{th} value against the $(i+1)^{th}$ value etc.

(see Gregory, Chapter 5)





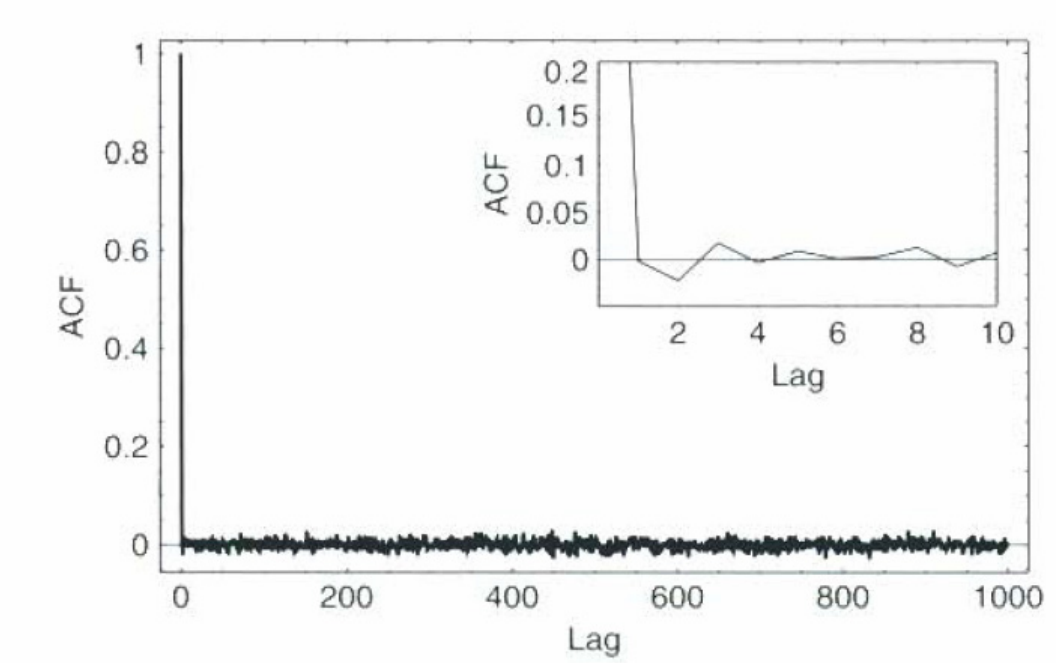
(b) Correlations between neighbouring pseudo-random numbers

Sequential patterns in the sampled values would show up as structure in the phase portraits – scatterplots of the i^{th} value against the $(i+1)^{th}$ value etc.

We can compute the Auto-correlation function

$$\rho(j) = \frac{\sum\left[(x_i - \bar{x})(x_{i+j} - \bar{x})\right]}{\sqrt{\sum (x_i - \bar{x})^2}\sqrt{\sum (x_{i+j} - \bar{x})^2}}$$

j is known as the Lag



If the sequence is uniformly random, we expect

$$\begin{cases} \rho(j) = 1 & \text{for } j = 0 \\ \rho(j) = 0 & \text{otherwise} \end{cases}$$

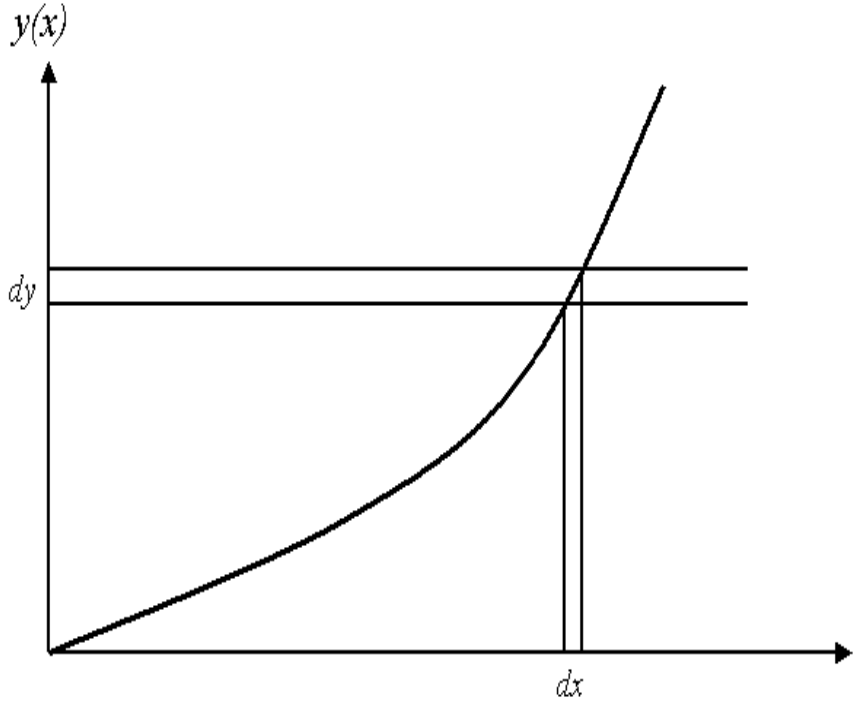
6.2. Variable transformations

Generating random numbers from other pdfs can be done by transforming random numbers drawn from simpler pdfs.

The procedure is similar to changing variables in integration.

Suppose, e.g. $x \sim p(x)$

Let $y = y(x)$ be **monotonic**



Then $$p(y)dy = p(x)dx$$

Probability of number between y and $y+dy$

Probability of number between x and $x+dx$

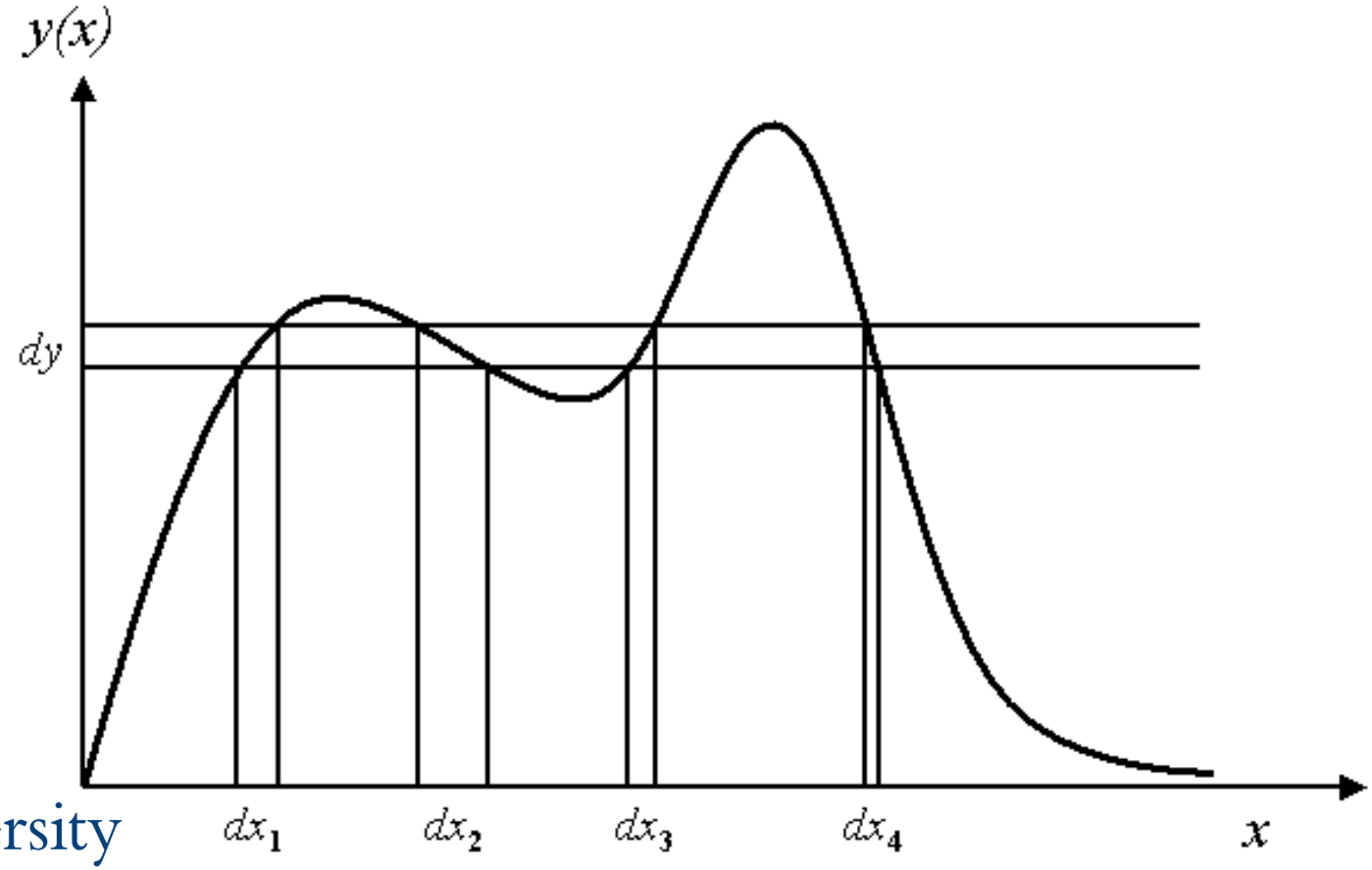
$$p(y) = \frac{p(x(y))}{|dy/dx|}$$

Because probability must be positive

We can extend the expression given previously to the case where $y(x)$ is not monotonic, by calculating

$$p(y)dy = \sum_i p(x_i)dx_i \quad \text{so that}$$

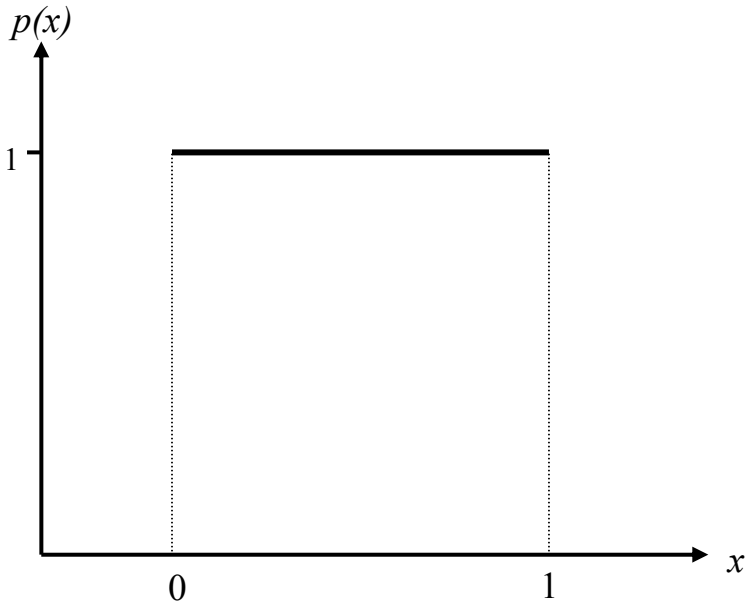
$$p(y) = \sum_i \frac{p(x_i(y))}{\left|dy/dx_i\right|}$$



Example 1

Suppose we have $x \sim U[0,1]$

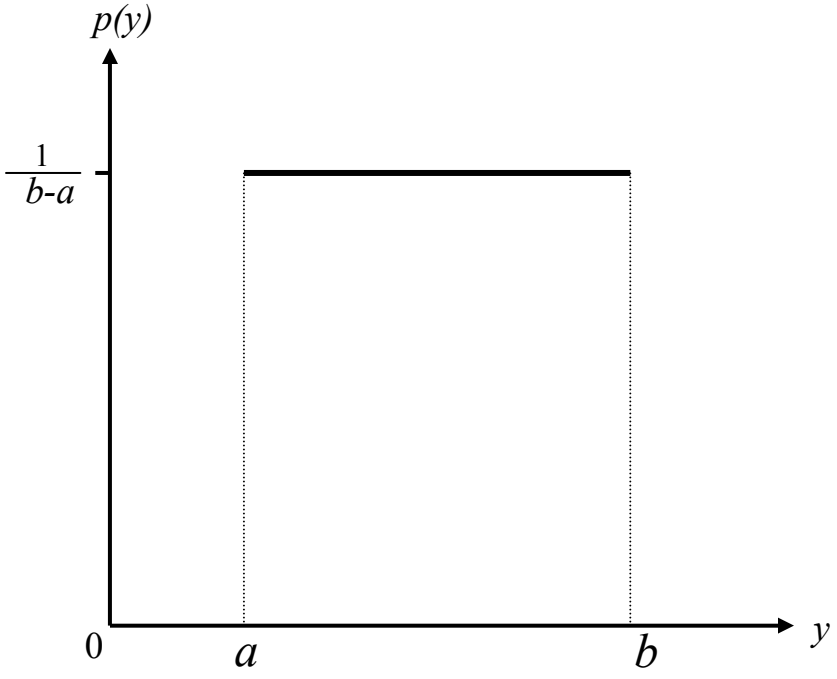
Then $p(x) = \begin{cases} 1 & \text{for} \quad 0 < x < 1 \\ 0 & \text{otherwise} \end{cases}$



Define $y = a + (b-a)x$

So $\frac{dy}{dx} = (b-a)$

i.e. $p(y) = \begin{cases} 1/b-a & \text{for} \quad a < y < b \\ 0 & \text{otherwise} \end{cases}$



or $y \sim U[a,b]$

Example 2

Normal pdf with mean zero and standard deviation unity

Numerical recipes provides a program to turn $x \sim U[0,1]$ into $y \sim N[0,1]$

Suppose we want $z \sim N[\mu, \sigma]$

We define $z = \mu + \sigma y$ so that $\frac{dz}{dy} = \sigma$

Now $$p(y) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2} y^2\right)$$

so $$p(z) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left[\frac{z-\mu}{\sigma}\right]^2\right)$$

Normal pdf with mean zero and standard deviation unity

Example 2

Numerical recipes provides a program to turn $x \sim U[0,1]$ into $y \sim N[0,1]$

Suppose we want $z \sim N[\mu, \sigma]$

We define $z = \mu + \sigma y$ so that $\frac{dz}{dy} = \sigma$

Now $$p(y) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2} y^2\right)$$

so $$p(z) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left[\frac{z-\mu}{\sigma}\right]^2\right)$$

Variable transformation formula also the basis for 'error propagation' formulae we use in data analysis - see also SUPAIDA course

Question 13: If $x \sim U[0,1]$ and $y = -\ln x$, the pdf of y is

A $p(y) = e^{y}$

B $p(y) = e^{-y}$

C $p(y) = -\ln y$

D $p(y) = \ln y$

Question 13: If $x \sim U[0,1]$ and $y = -\ln x$, the pdf of y is

A $p(y) = e^{y}$

B $p(y) = e^{-y}$

C $p(y) = -\ln y$

D $p(y) = \ln y$

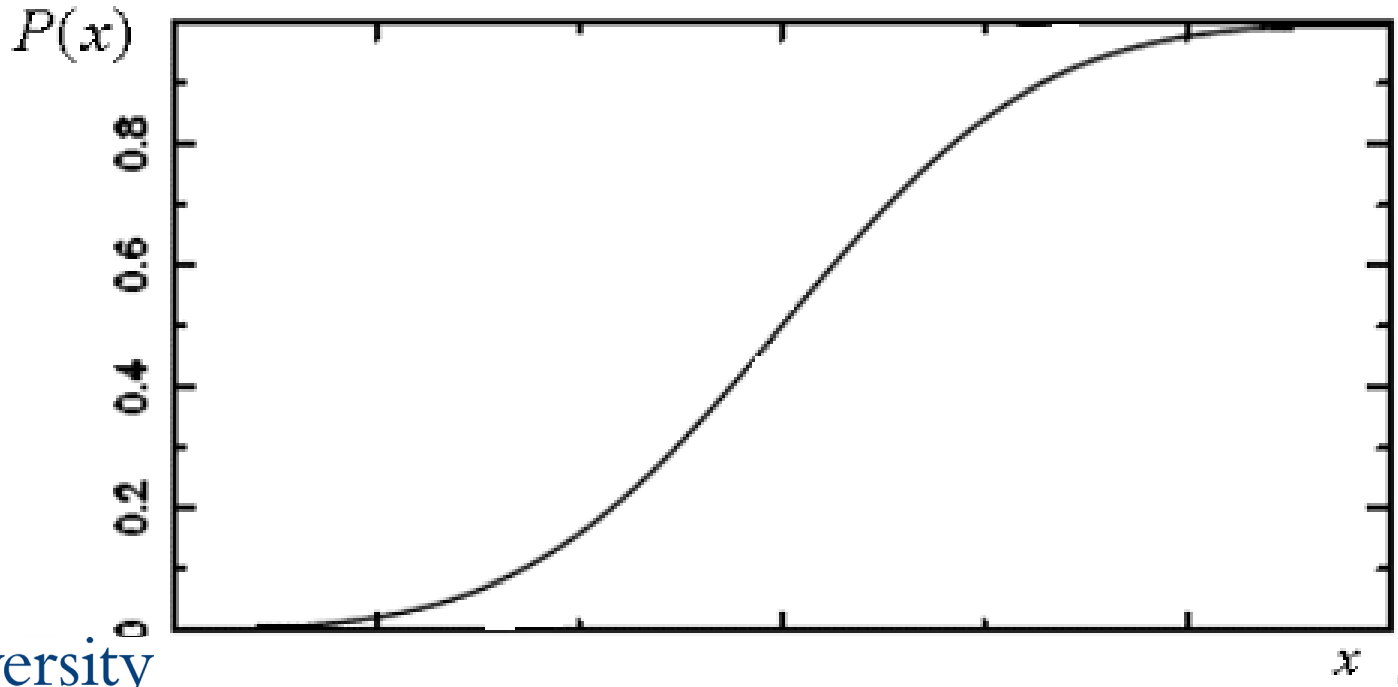
6.3. Probability integral transform

One particular variable transformation merits special attention.

Suppose we can compute the CDF of some desired random variable

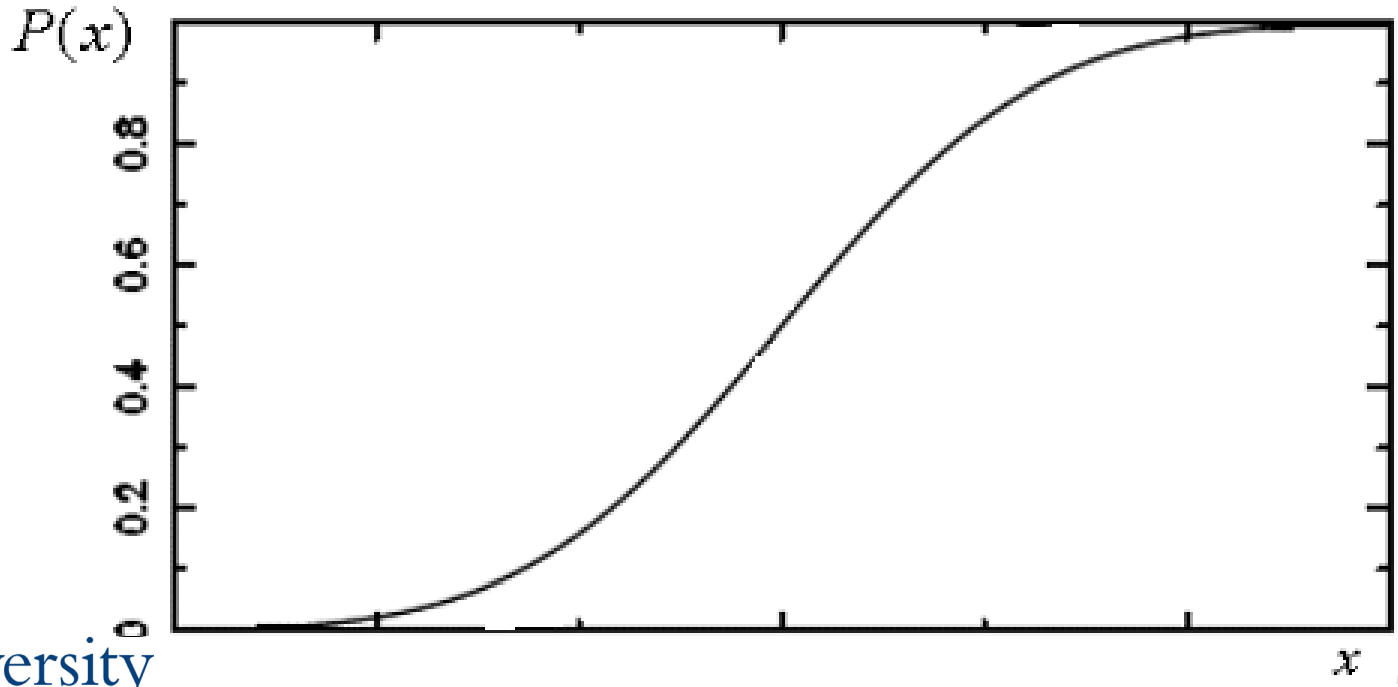
Cumulative distribution function (CDF)

$$P(a) = \int_{-\infty}^{a} p(x)dx = \text{Prob}(x < a)$$



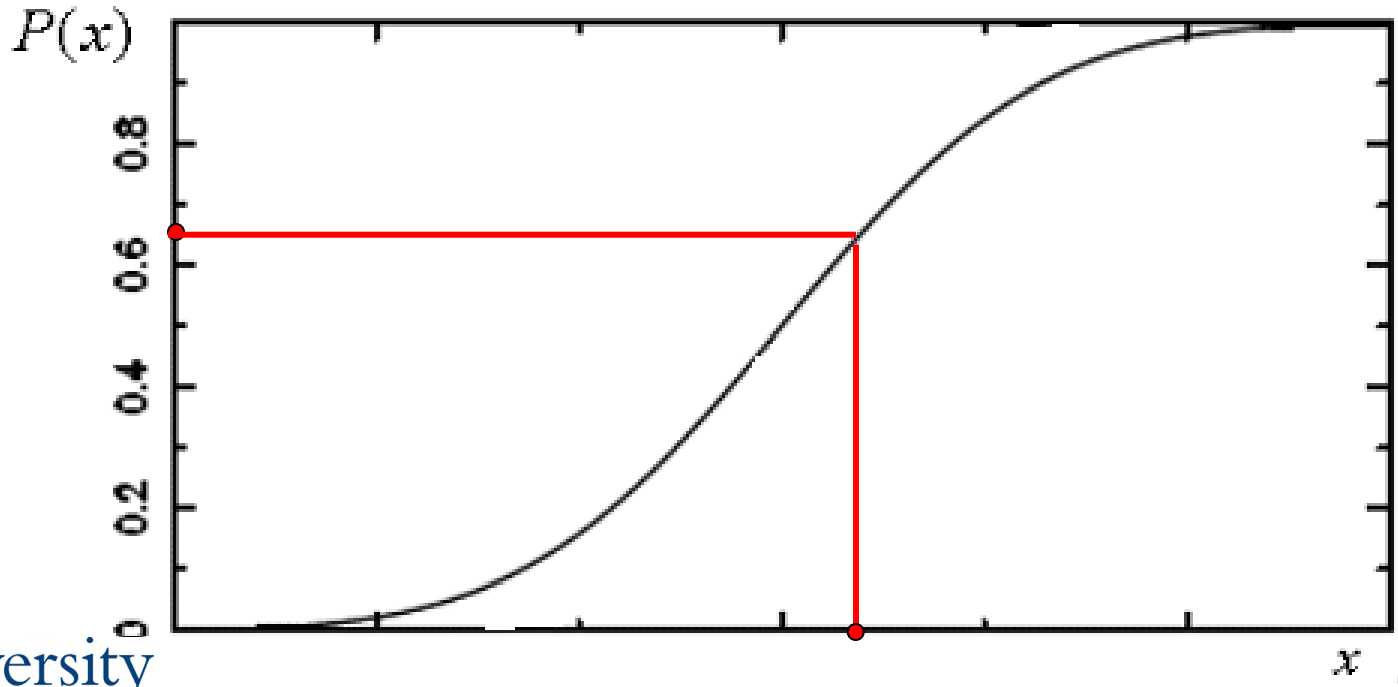
1) Sample a random variable $y \sim U[0,1]$

2) Compute x such that $y = P(x)$ i.e. $x = P^{-1}(y)$



1) Sample a random variable $y \sim U[0,1]$

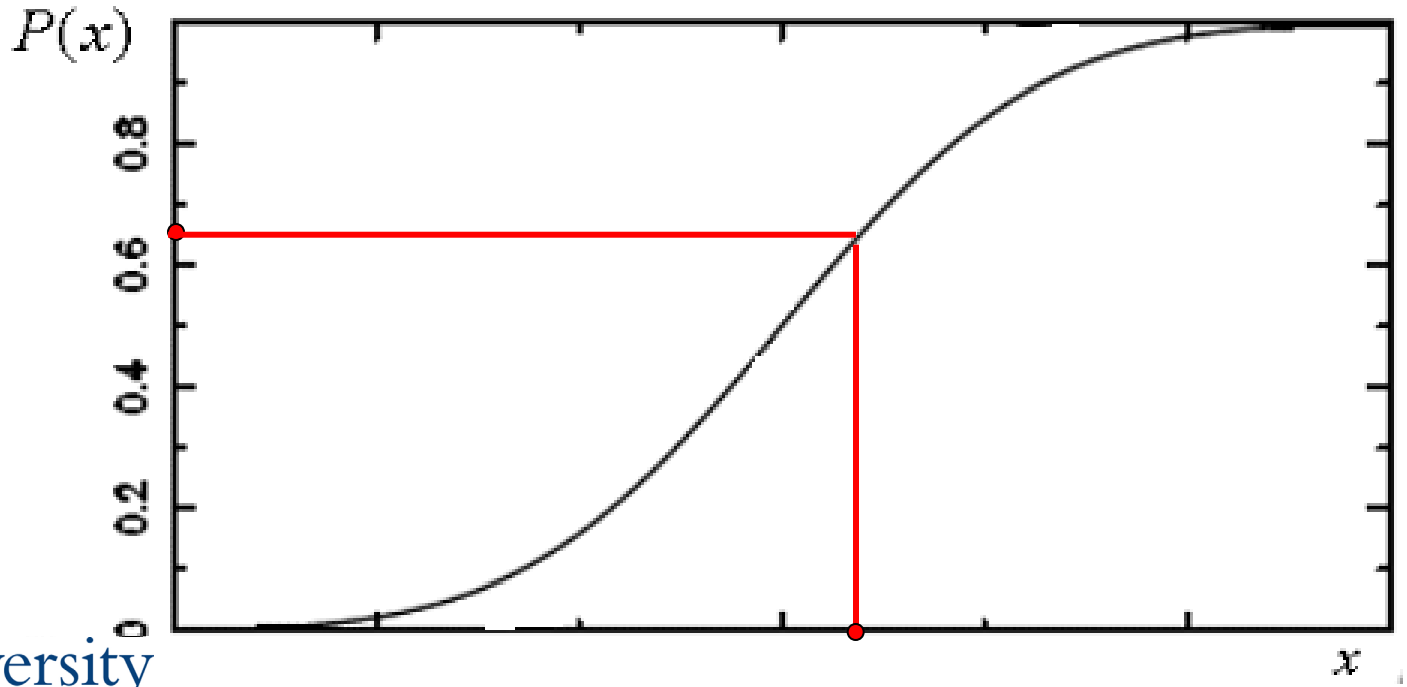
2) Compute x such that $y = P(x)$ i.e. $x = P^{-1}(y)$



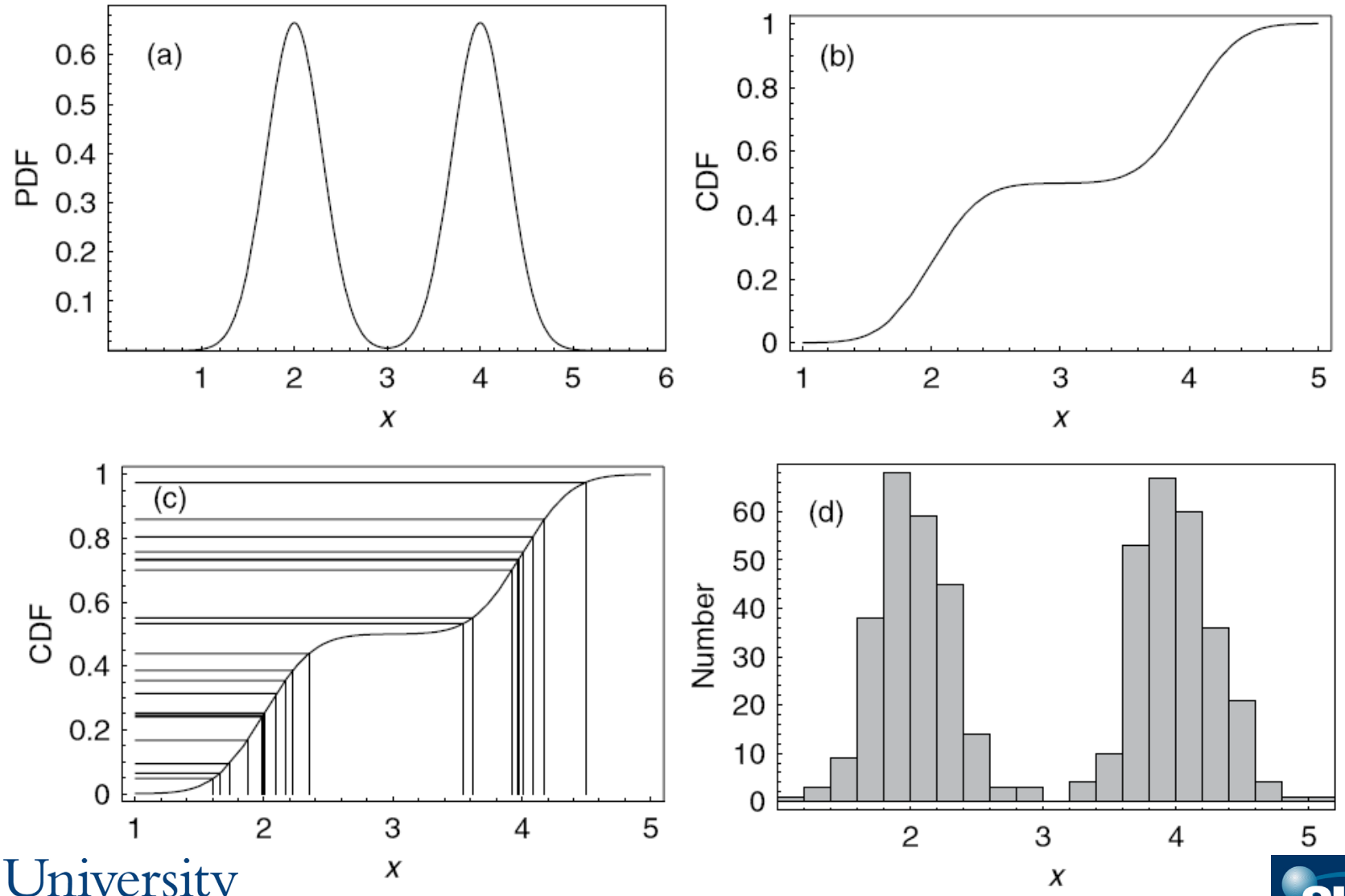
1) Sample a random variable $y \sim U[0,1]$

2) Compute x such that $y = P(x)$ i.e. $x = P^{-1}(y)$

3) Then $x \sim p(x)$ i.e. x is drawn from the pdf corresponding to the cdf $P(x)$



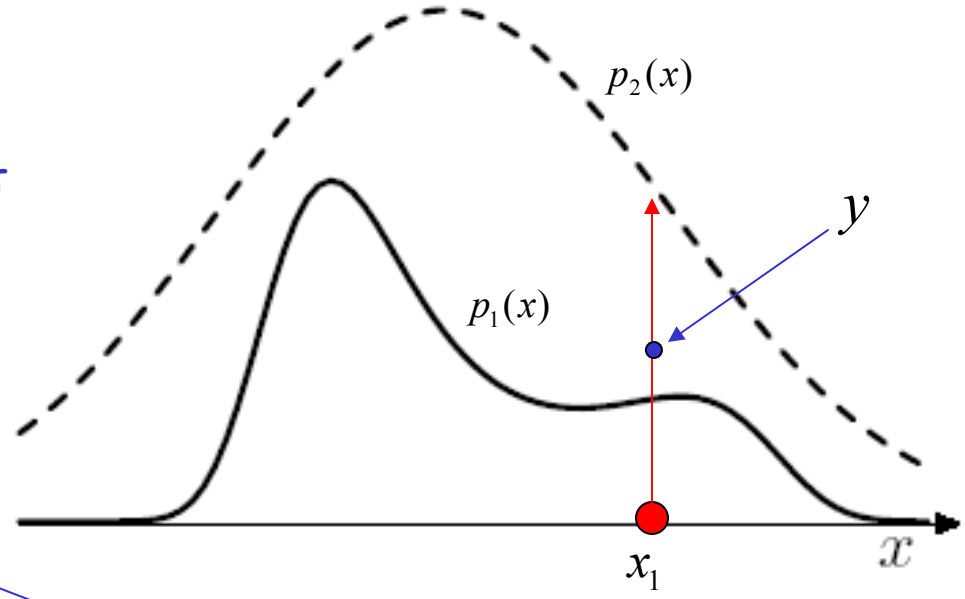
Example (from Gregory)



6.4. Rejection sampling

Suppose we want to sample from some pdf $p_1(x)$ and we know that

$$p_1(x) < p_2(x) \quad \forall x$$



1) Sample x_1 from $p_2(x)$

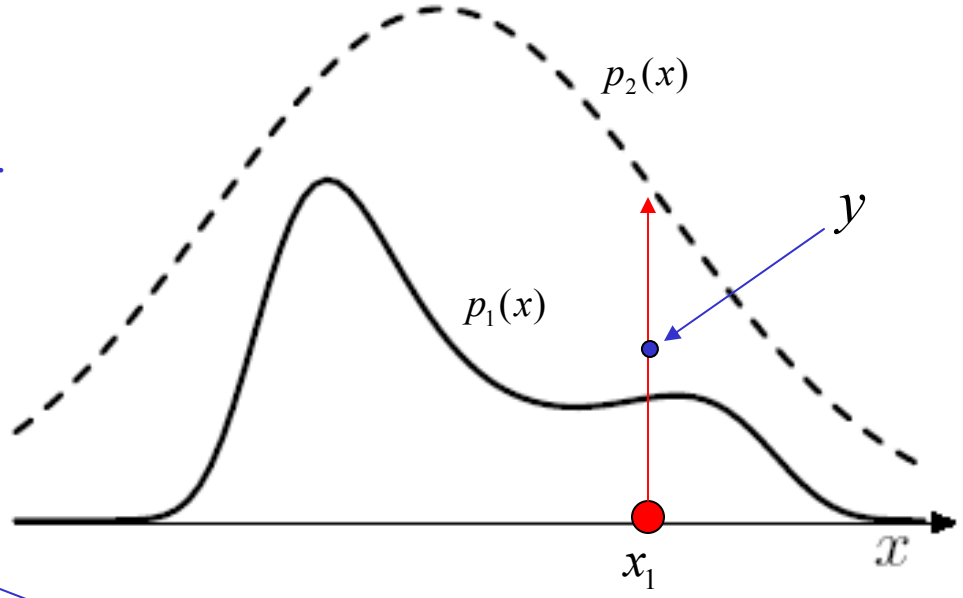
2) Sample $y \sim U[0, p_2(x_1)]$

(Suppose we have an 'easy' way to do this)

6.4. Rejection sampling

Suppose we want to sample from some pdf $p_1(x)$ and we know that

$$p_1(x) < p_2(x) \quad \forall x$$



1) Sample x_1 from $p_2(x)$

2) Sample $y \sim U[0, p_2(x_1)]$

(Suppose we have an 'easy' way to do this)

3) If $y < p_1(x)$ **ACCEPT**

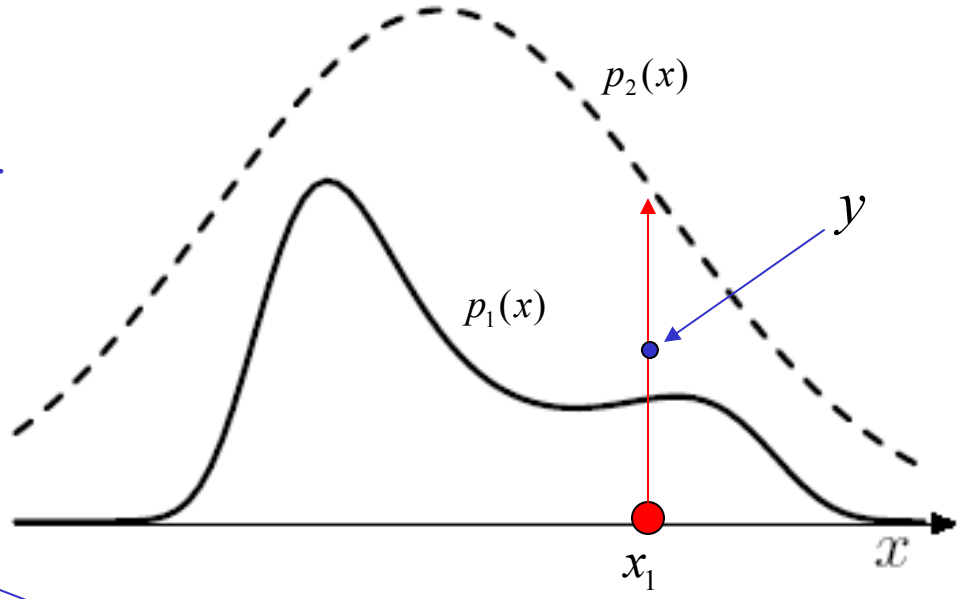
otherwise **REJECT**

6.4. Rejection sampling

(following Mackay)

Suppose we want to sample from some pdf $p_1(x)$ and we know that

$$p_1(x) < p_2(x) \quad \forall x$$



1) Sample x_1 from $p_2(x)$

2) Sample $y \sim U[0, p_2(x_1)]$

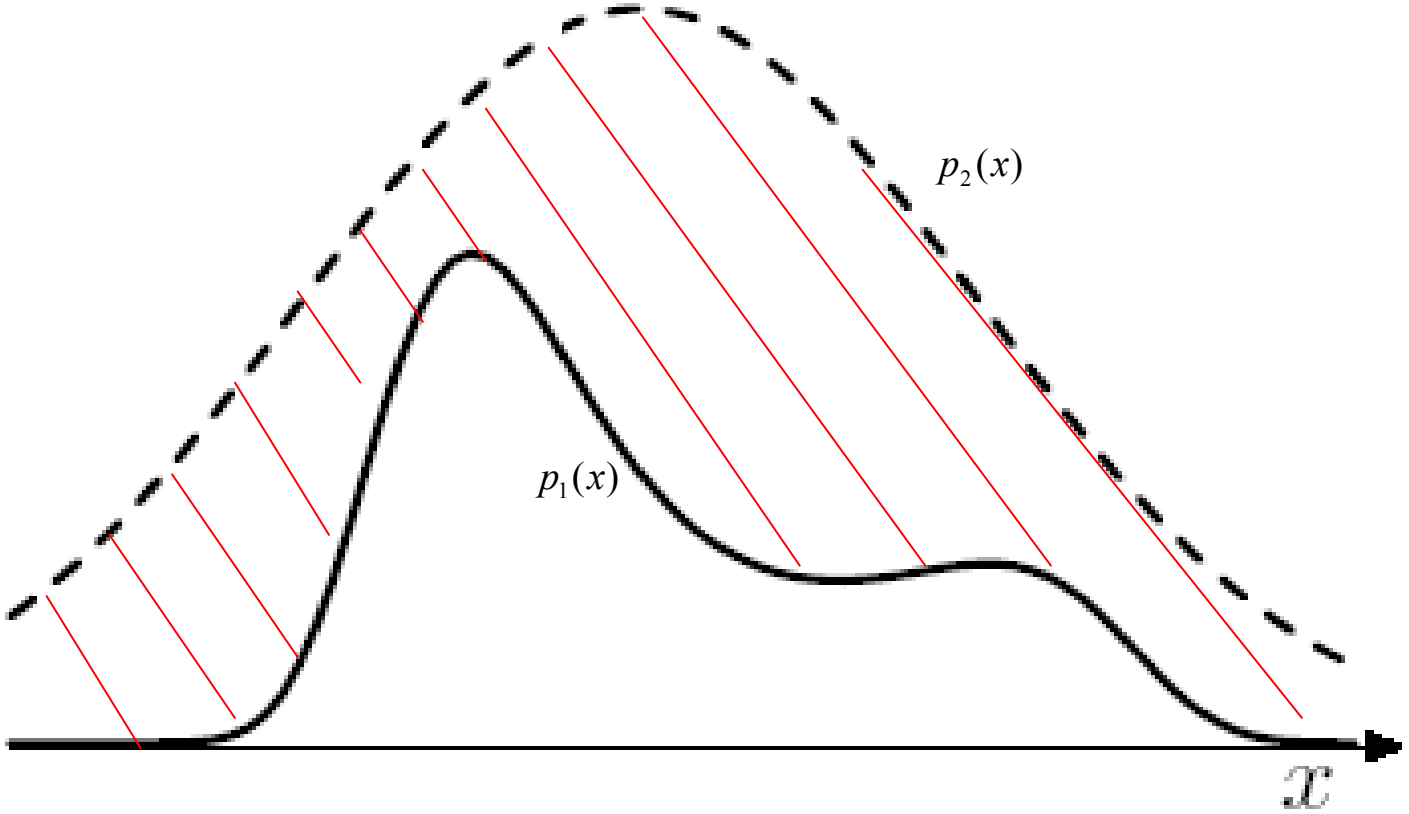
(Suppose we have an 'easy' way to do this)

3) If $y < p_1(x)$ **ACCEPT**

otherwise **REJECT**

Set of accepted values $\{x_i\}$ *are a sample from* $p_1(x)$

6.4. Rejection sampling



Method can be very slow if the shaded region is too large.

Ideally we want to find a pdf $p_2(x)$ that is: (a) easy to sample from

(b) close to $p_1(x)$

6.5. Genetic Algorithms

1995ApJS..101..309C

THE ASTROPHYSICAL JOURNAL SUPPLEMENT SERIES, 101:309–334, 1995 December

GENETIC ALGORITHMS IN ASTRONOMY AND ASTROPHYSICS

P. CHARBONNEAU
High Altitude Observatory, National Center for Atmospheric Research,[1] P.O. Box 3000, Boulder, CO 80307-3000;
paulchar@hao.ucar.edu
Received 1994 December 30; accepted 1995 June 9

ABSTRACT

This paper aims at demonstrating, through examples, the applicability of *genetic algorithms* to wide classes of problems encountered in astronomy and astrophysics. Genetic algorithms are heuristic search techniques that incorporate, in a computational setting, the biological notion of evolution by means of natural selection. While increasingly in use in the fields of computer science, artificial intelligence, and computed-aided engineering design, genetic algorithms seem to have attracted comparatively little attention in the physical sciences thus far.

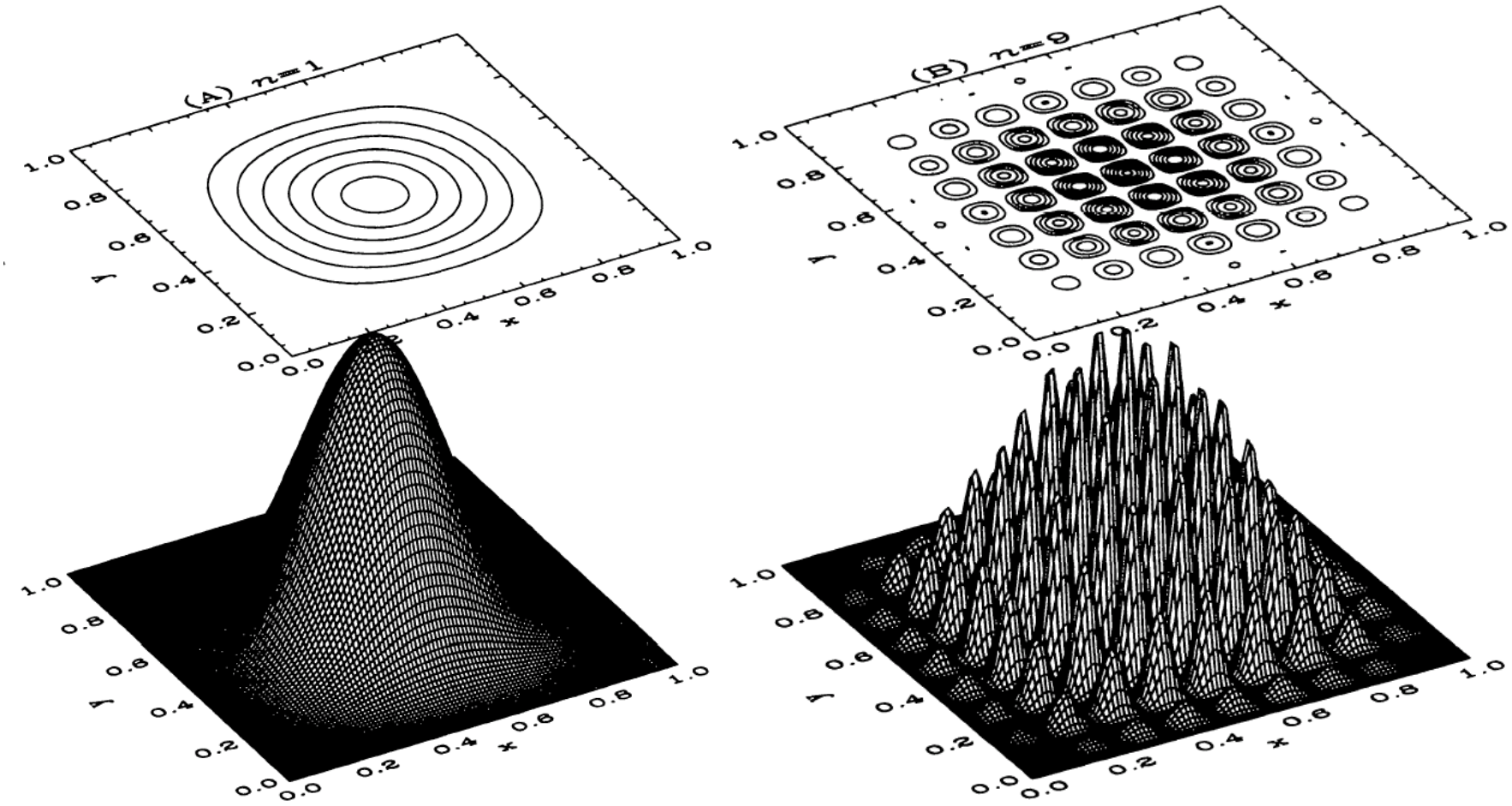
The following three problems are treated: (1) modeling the rotation curve of galaxies, (2) extracting pulsation periods from Doppler velocities measurements in spectral lines of δ Scuti stars, and (3) constructing spherically symmetric wind models for rotating, magnetized solar-type stars. A listing of the genetic algorithm-based general purpose optimization subroutine `PIKAIA`, used to solve these problems, is given in the Appendix.

Subject headings: galaxies: kinematics and dynamics — methods: numerical — stars: mass loss — stars: oscillations

6.5. Genetic Algorithms

$$f(x, y) = [16x(1-x)y(1-y)\sin(n\pi x)\sin(n\pi y)]^2,$$
$$x, y \in [0, 1], \quad n = 1, 2, \ldots$$

(Charbonneau 1995)



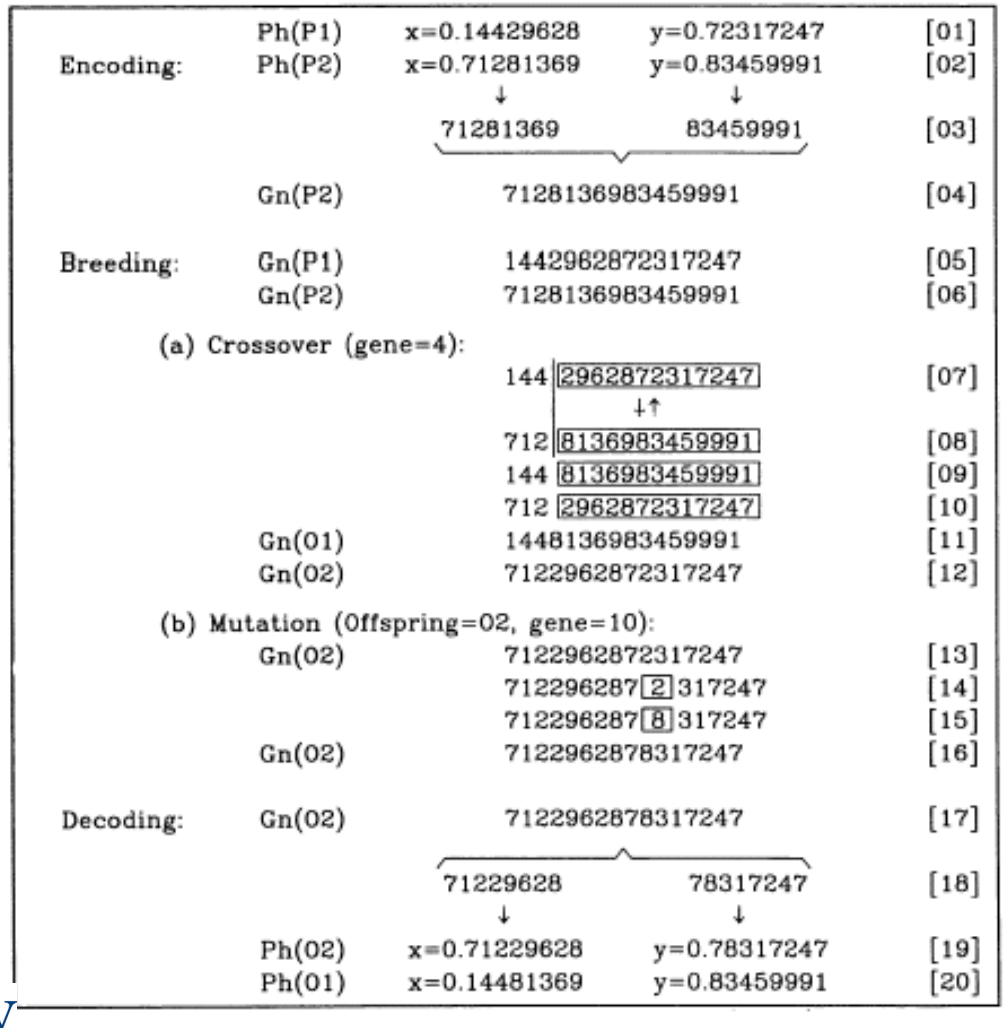
6.5. Genetic Algorithms

(Charbonneau 1995)

1. Construct a random initial population and evaluate the fitness of its members.
2. Construct a new population by breeding selected individuals from the old population.
3. Evaluate the fitness of each member of the new population.
4. Replace the old population by the new population.
5. Test convergence; unless fittest phenotype matches target phenotype within tolerance, `goto` step 2.

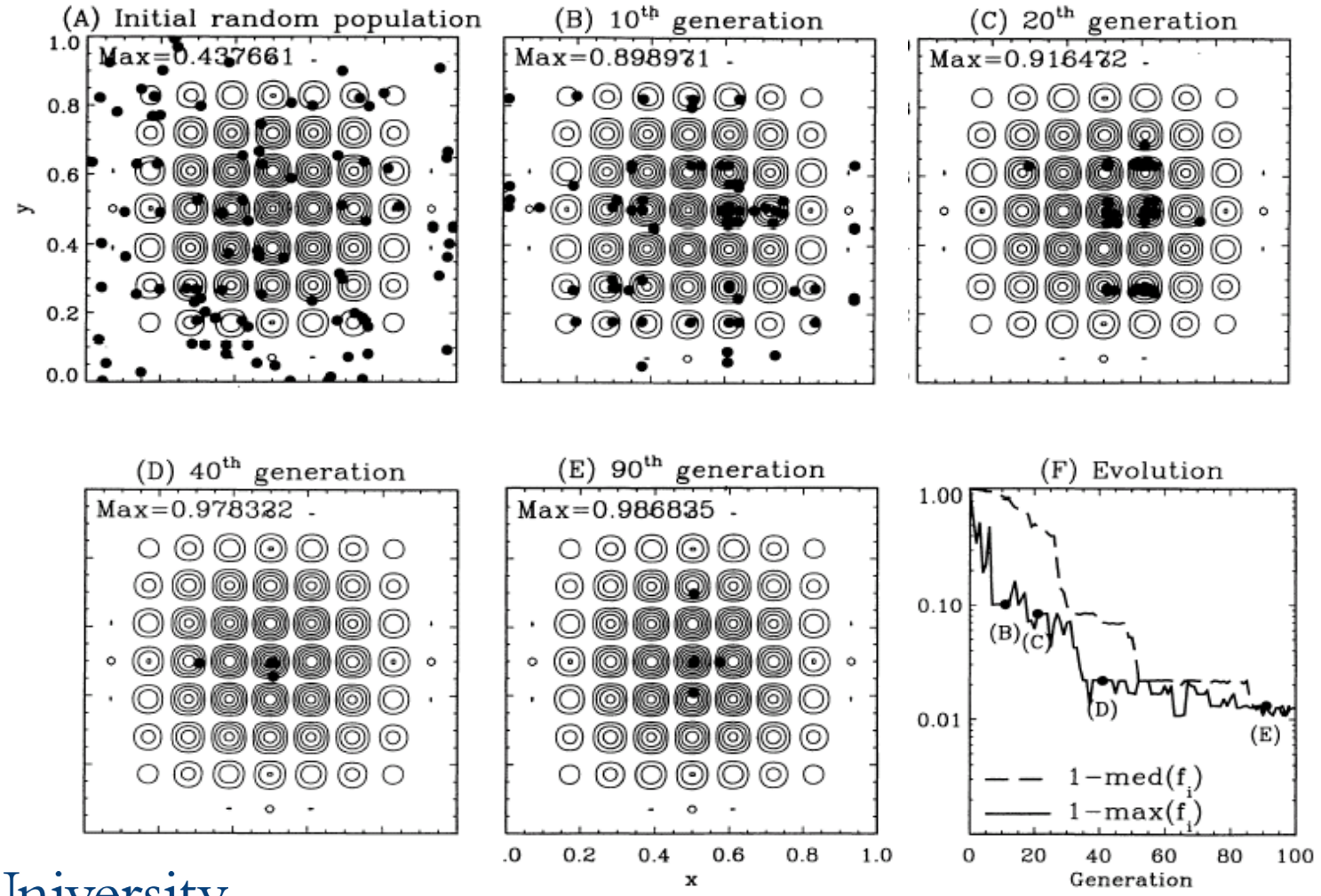
6.5. Genetic Algorithms

see http://www.hao.ucar.edu/Public/models/pikaia/pikaia.html



6.5. Genetic Algorithms

see http://www.hao.ucar.edu/Public/models/pikaia/pikaia.html

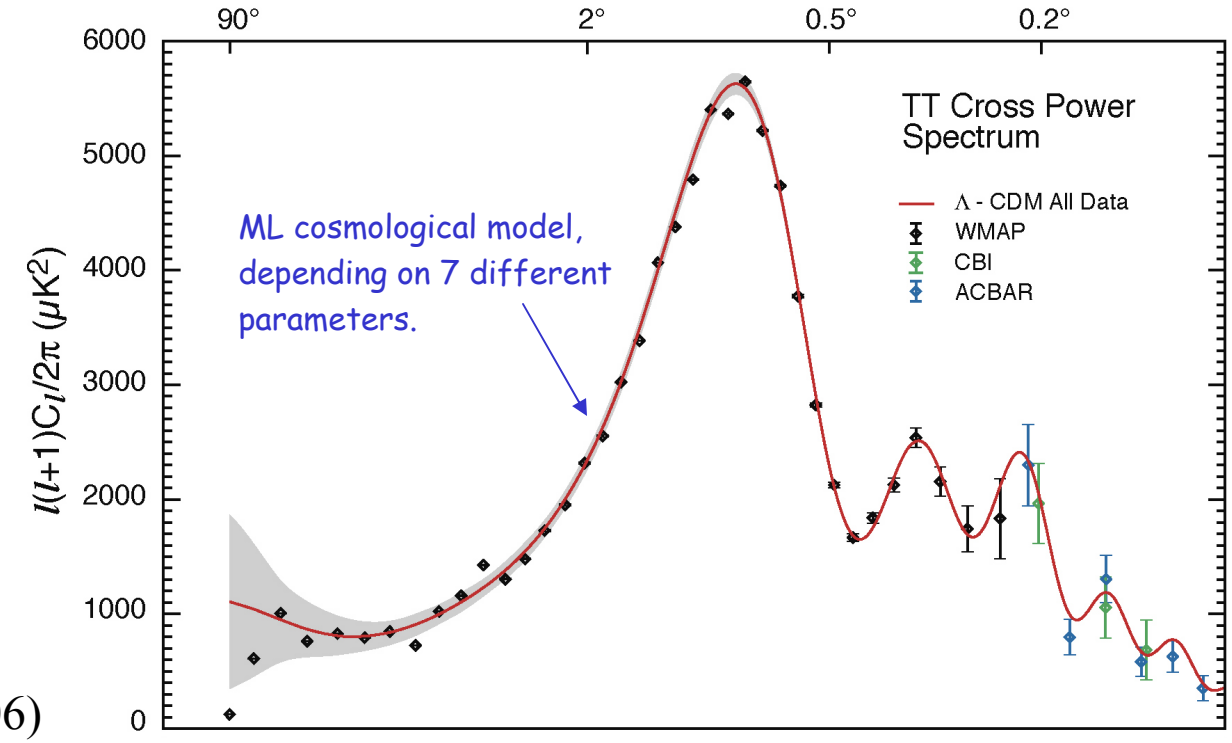


6.6. Markov Chain Monte Carlo

This is a very powerful, new (at least in astronomy!) method for sampling from pdfs. (These can be complicated and/or of high dimension).

MCMC widely used e.g. in cosmology to determine 'maximum likelihood' model to CMBR data.

Angular power spectrum of CMBR temperature fluctuations



(Hinshaw et al 2006)

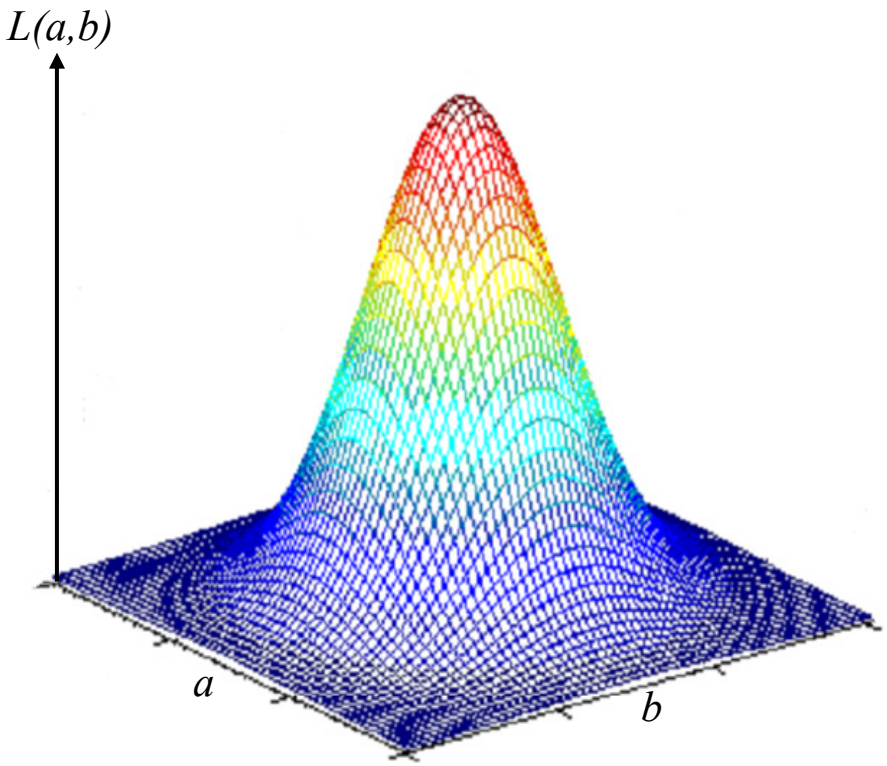
Consider a 2-D example (e.g. bivariate normal distribution);
Likelihood function depends on parameters a and b.

Suppose we are trying to find the maximum of $L(a,b)$

1) Start off at some randomly chosen value (a_1, b_1)

2) Compute $L(a_1, b_1)$ and gradient $\left(\frac{\partial L}{\partial a}, \frac{\partial L}{\partial b}\right)_{(a_1,b_1)}$

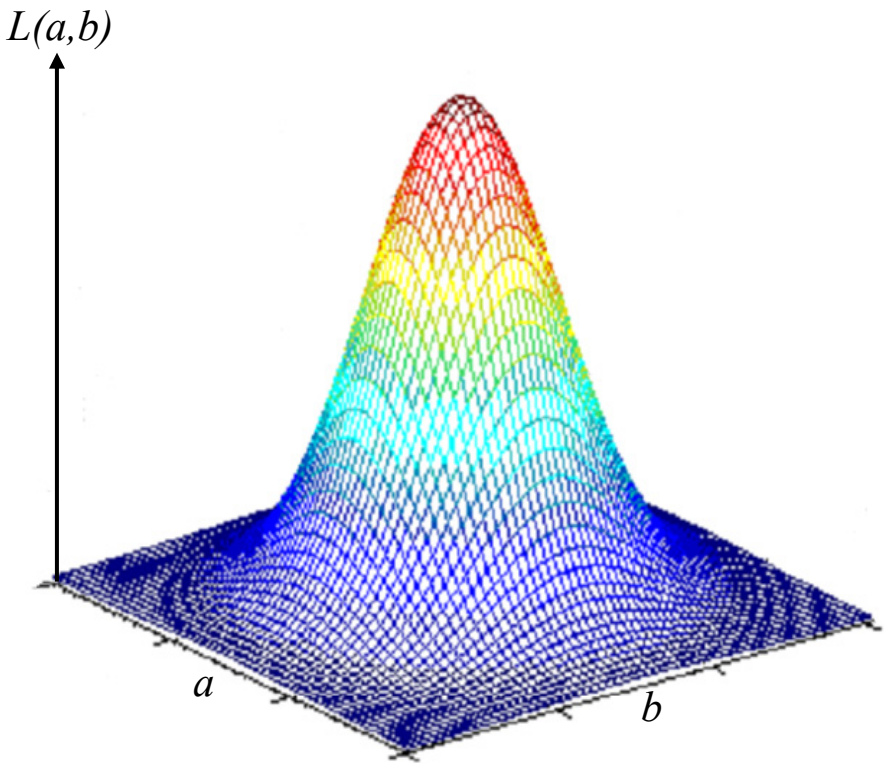
3) Move in direction of steepest +ve gradient – i.e. $L(a_1, b_1)$ is increasing fastest



4) Repeat from step 2 until (a_n, b_n) converges on maximum of likelihood

Consider a 2-D example (e.g. bivariate normal distribution);
Likelihood function depends on parameters a and b.

Suppose we are trying to find the maximum of $L(a,b)$



1) Start off at some randomly chosen value (a_1, b_1)

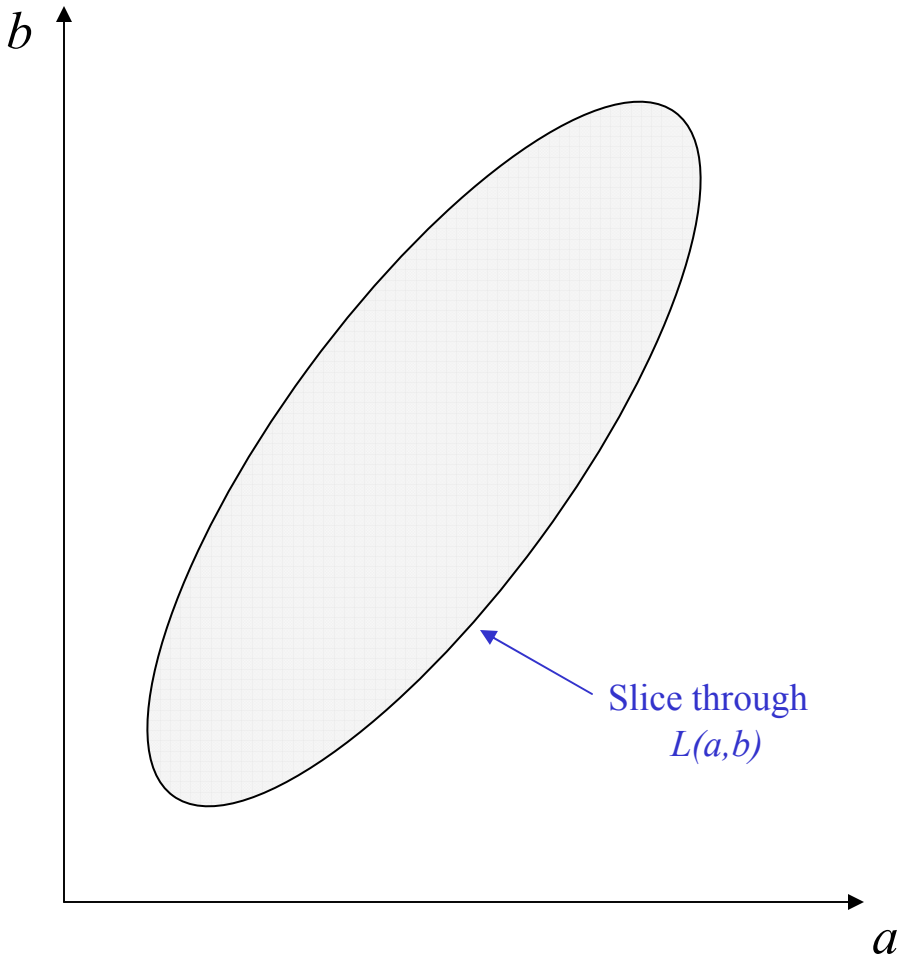
2) Compute $L(a_1, b_1)$ and gradient $\left(\frac{\partial L}{\partial a}, \frac{\partial L}{\partial b}\right)_{(a_1,b_1)}$

3) Move in direction of steepest +ve gradient – i.e. $L(a_1, b_1)$ is increasing fastest

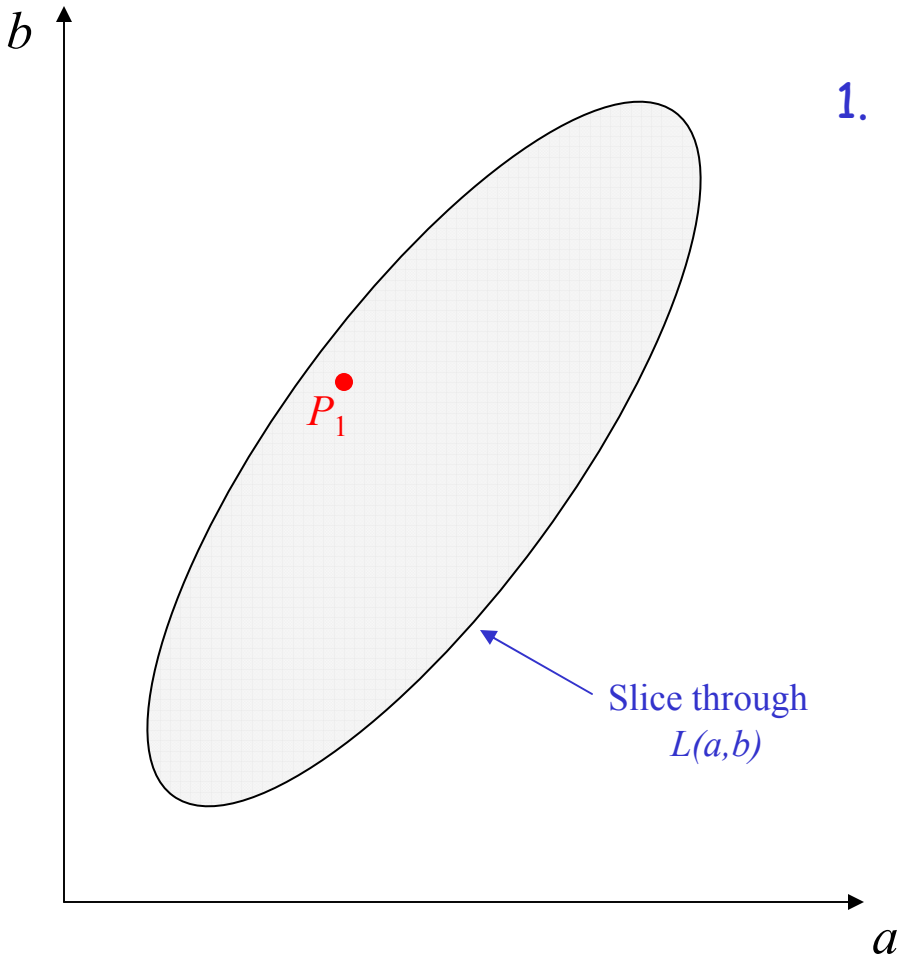
4) Repeat from step 2 until (a_n, b_n) converges on maximum of likelihood

OK for finding maximum, but not for generating a sample from $L(a,b)$ or for determining errors on the the ML parameter estimates.

MCMC provides a simple **Metropolis algorithm** for generating random samples of points from $L(a,b)$

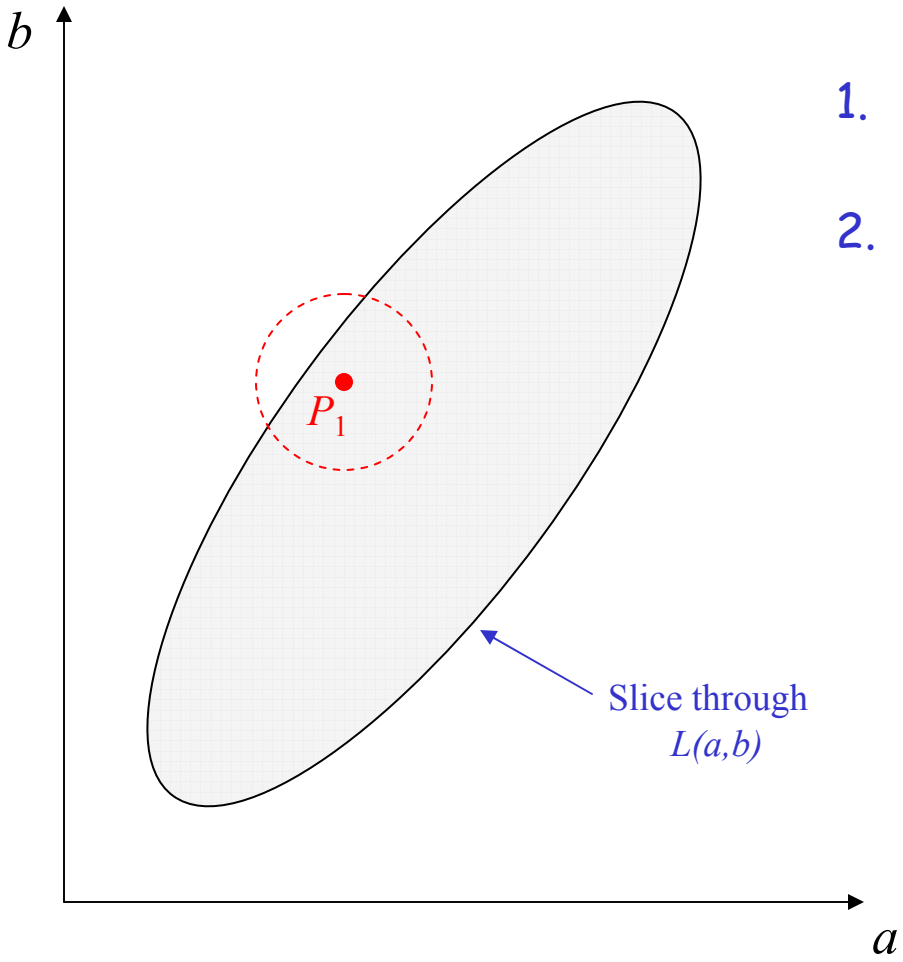


MCMC provides a simple **Metropolis algorithm** for generating random samples of points from $L(a,b)$



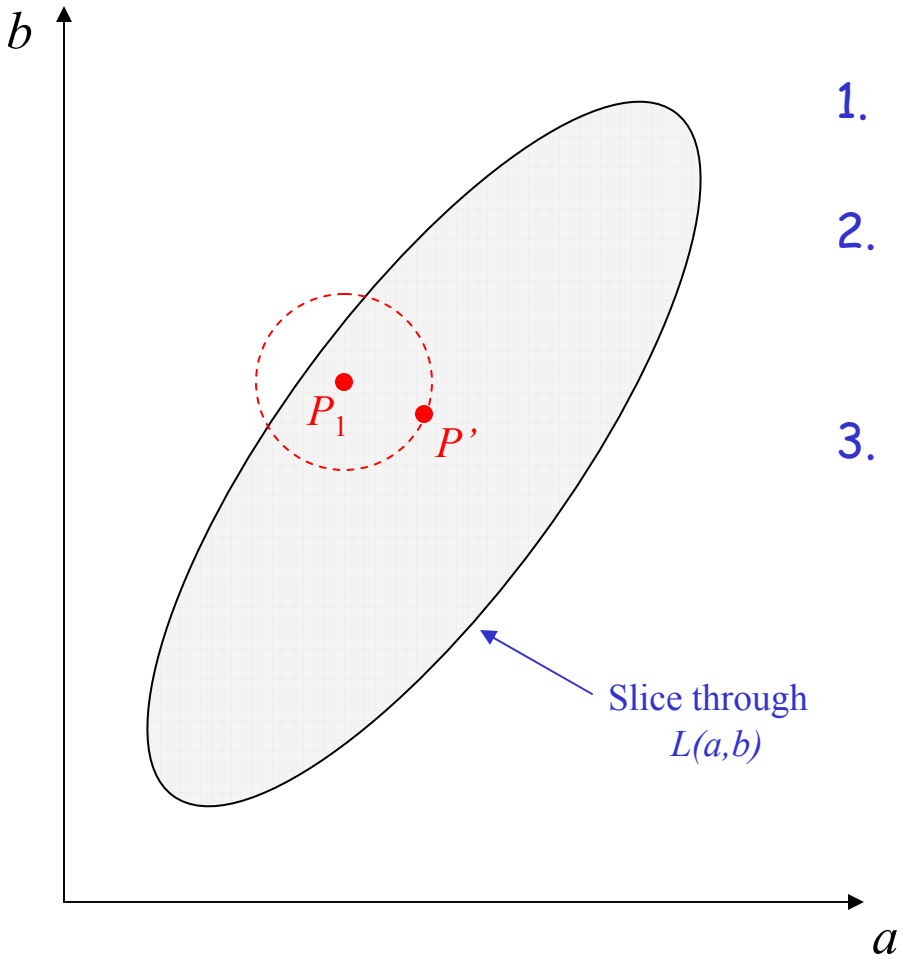
1. Sample random initial point $P_1 = (a_1, b_1)$

MCMC provides a simple **Metropolis algorithm** for generating random samples of points from $L(a,b)$



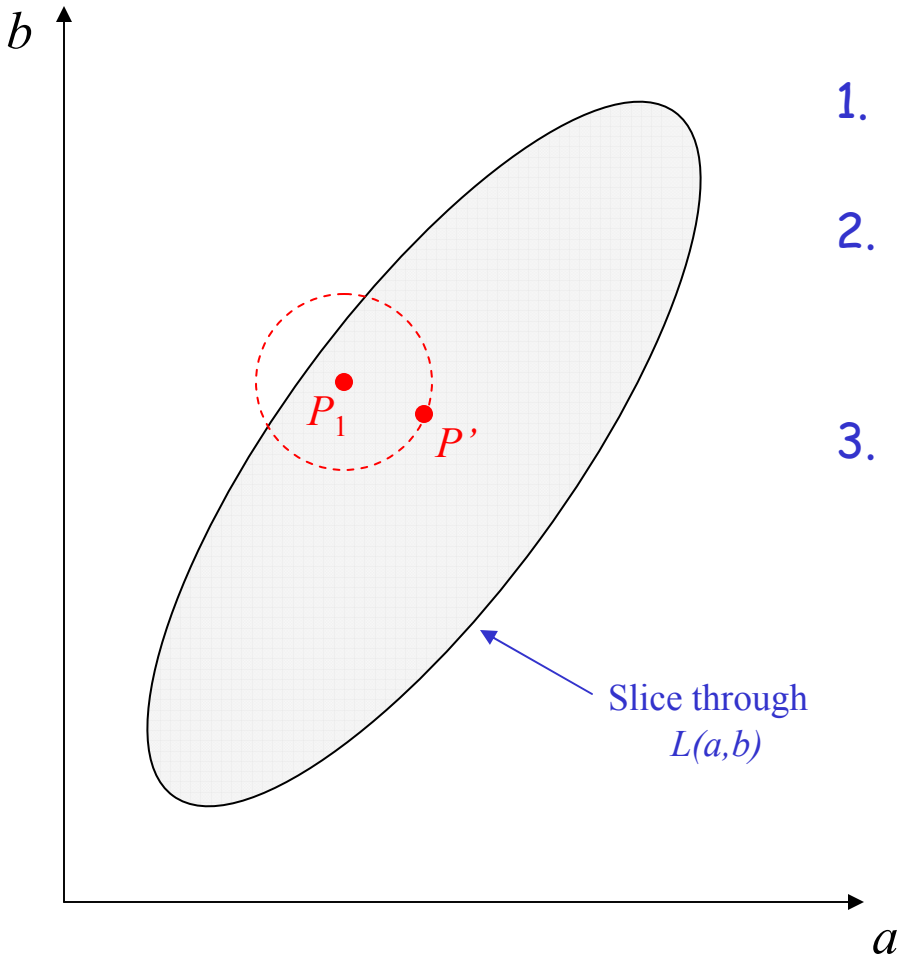
1. Sample random initial point $P_1 = (a_1, b_1)$
2. Centre a new pdf, Q, called the **proposal density**, on P_1

MCMC provides a simple **Metropolis algorithm** for generating random samples of points from $L(a,b)$



1. Sample random initial point $P_1 = (a_1, b_1)$
2. Centre a new pdf, Q, called the **proposal density**, on P_1
3. Sample tentative new point $P' = (a', b')$ from Q

MCMC provides a simple **Metropolis algorithm** for generating random samples of points from $L(a,b)$



1. Sample random initial point $P_1 = (a_1, b_1)$
2. Centre a new pdf, Q, called the **proposal density**, on P_1
3. Sample tentative new point $P' = (a', b')$ from Q
4. Compute $$R = \frac{L(a',b')}{L(a_1,b_1)}$$

5. If $R > 1$ this means P' is **uphill** from P_1.

We **accept** P' as the next point in our chain, i.e. $P_2 = P'$

6. If $R < 1$ this means P' is **downhill** from P_1.

In this case we ***may*** reject P' as our next point.

In fact, we accept P' with probability R.

How do we do this?...

(a) Generate a random number $x \sim \mathrm{U}[0,1]$

(b) If $x < R$ then accept P' and set $P_2 = P'$

(c) If $x > R$ then reject P' and set $P_2 = P_1$

5. If $R > 1$ this means P' is **uphill** from P_1.

We **accept** P' as the next point in our chain, i.e. $P_2 = P'$

6. If $R < 1$ this means P' is **downhill** from P_1.

In this case we ***may*** reject P' as our next point.

In fact, we accept P' with probability R.

How do we do this?...

(a) Generate a random number $x \sim \mathrm{U}[0,1]$

(b) If $x < R$ then accept P' and set $P_2 = P'$

(c) If $x > R$ then reject P' and set $P_2 = P_1$

Acceptance probability depends only on the previous point - ***Markov Chain***

So the Metropolis Algorithm generally (but not always) moves uphill, towards the peak of the Likelihood Function.

Remarkable facts

- Sequence of points $\{ P_1, P_2, P_3, P_4, P_5, \ldots \}$ represents a sample from the LF $L(a,b)$ (see notes on website)
- Sequence for each coordinate, e.g. $\{ a_1, a_2, a_3, a_4, a_5, \ldots \}$ samples the marginalised likelihood of a
- We can make a histogram of $\{ a_1, a_2, a_3, a_4, a_5, \ldots, a_n \}$ and use it to compute the mean and variance of a (i.e. to attach an error bar to a)

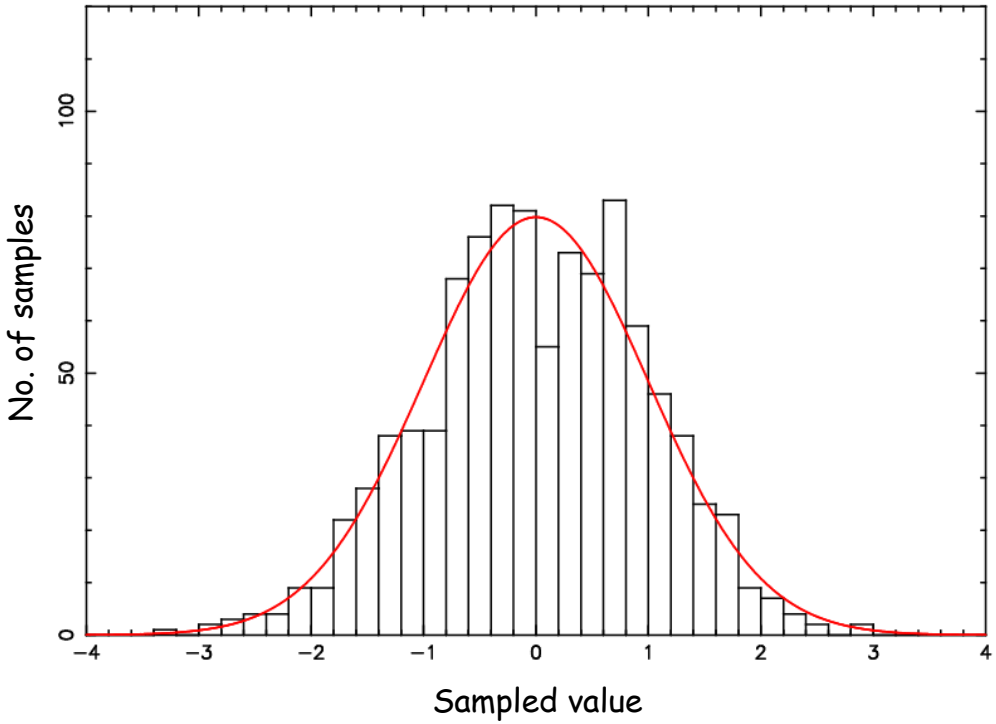
Why is this so useful?…

Suppose our LF was a 1-D Gaussian. We could estimate the mean and variance quite well from a histogram of e.g. 1000 samples.

But what if our problem is, e.g. 7 dimensional?

'Exhaustive' sampling could require $(1000)^7$ samples!

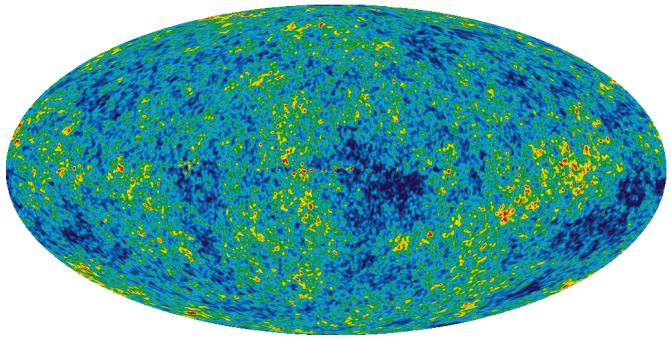
MCMC provides a short-cut.

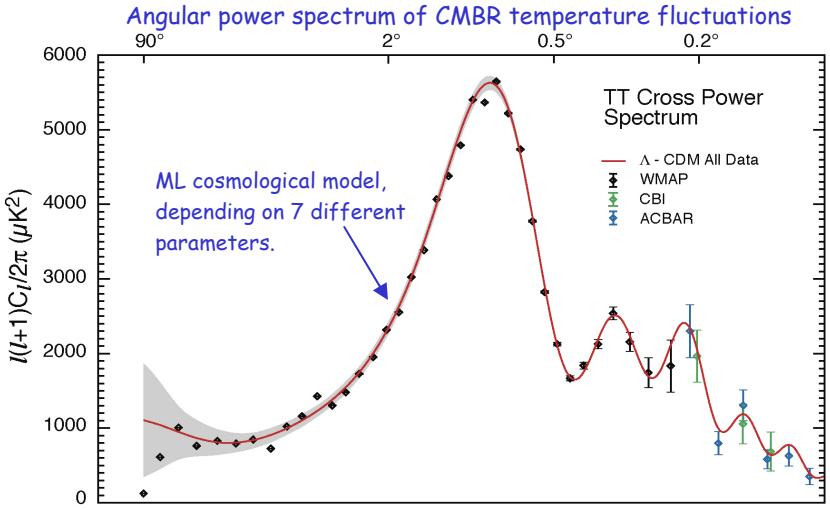


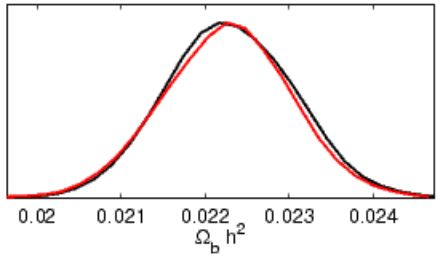
To compute a new point in our Markov Chain we need to compute the LF. But the computational cost does **not** grow so dramatically as we increase the number of dimensions of our problem.

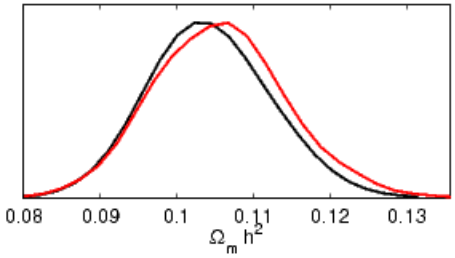
This lets us tackle problems that would be impossible by 'normal' sampling.

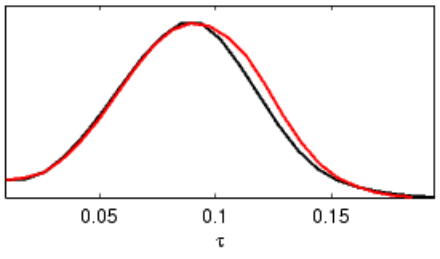
Example: CMBR constraints from WMAP 3 year data (+ 1 year data)

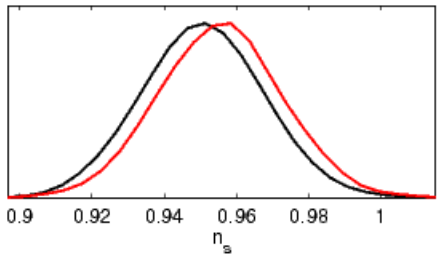


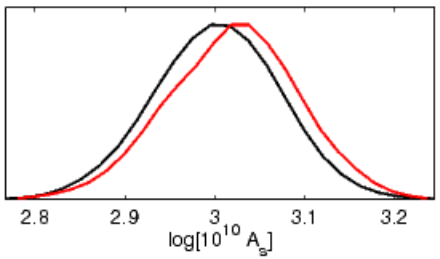


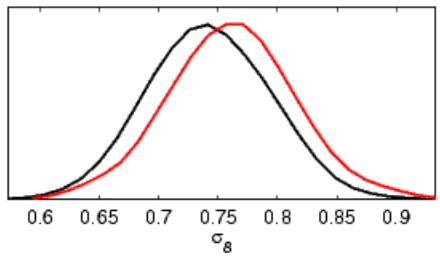


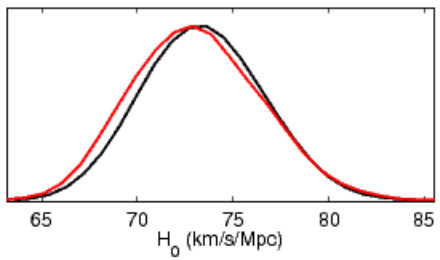












(Hinshaw et al 2006)

Question 14: When applying the Metropolis algorithm, if the width of the proposal density is very small

A the Markov Chain will move around the parameter space very slowly

B the Markov Chain will converge very quickly to the true pdf

C the acceptance rate of proposed steps in the Markov Chain will be very small

D most steps in the Markov Chain will explore regions of very low probability

Question 14: When applying the Metropolis algorithm, if the width of the proposal density is very small

A the Markov Chain will move around the parameter space very slowly

B the Markov Chain will converge very quickly to the true pdf

C the acceptance rate of proposed steps in the Markov Chain will be very small

D most steps in the Markov Chain will explore regions of very low probability

A number of factors can improve the performance of the Metropolis algorithm, including:

- using parameters in the likelihood function which are (close to) independent (i.e. their Fisher matrix is approx. diagonal).

- adopting a judicious choice of proposal density, well matched to the shape of the likelihood function.

- using a **simulated annealing** approach – i.e. sampling from a modified posterior likelihood function of the form

$$p_T(\theta \mid D, I) = \exp\left\{\frac{\ln\left[p(\theta \mid D, I)\right]}{T}\right\}$$

for large T the modified likelihood is a flatter version of the true likelihood

Temperature parameter T starts out large, so that the acceptance rate for 'downhill' steps is high – search is essentially random.

(This helps to avoid getting stuck in local maxima)

T is gradually reduced as the chain evolves, so that 'downhill' steps become increasingly disfavoured.

In some versions, the evolution of T is carried out automatically – this is known as adaptive simulated annealing.

See, for example, Numerical Recipes Section 10.9, or Gregory Chapter 11, for more details.

A related idea is **parallel tempering** (see e.g. Gregory, Chap 12)

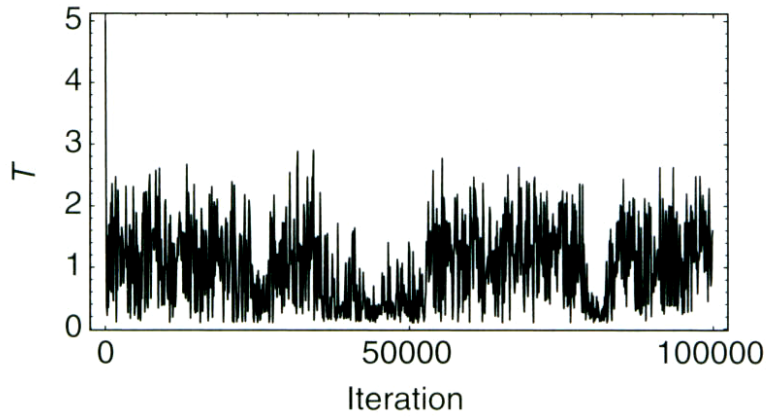
Series of MCMC chains, with different $\beta = 1/T,$ set off in parallel, with a certain probability of swapping parameter states between chains.

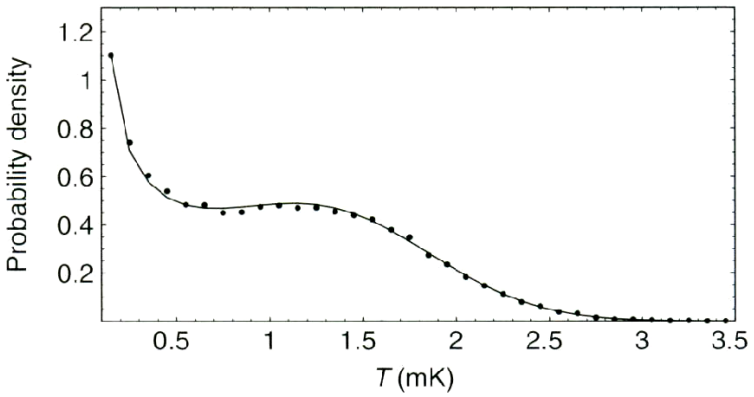
High temperature chains are effective at mapping out the global structure of the likelihood surface.

Low temperature chains are effective at mapping out the shape of local likelihood maxima.

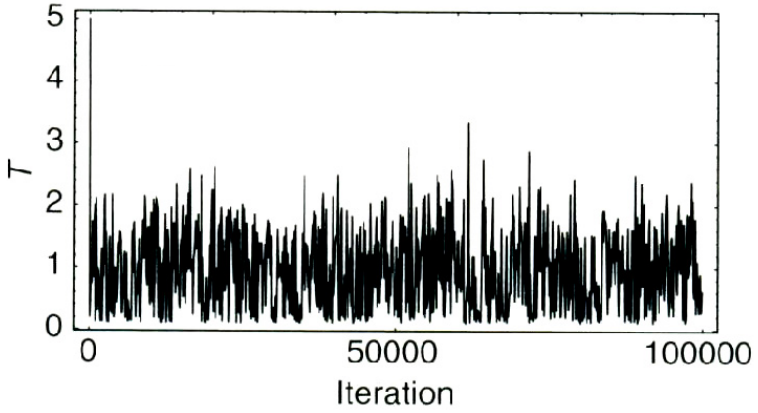
Example: spectral line fitting, from Section 3.

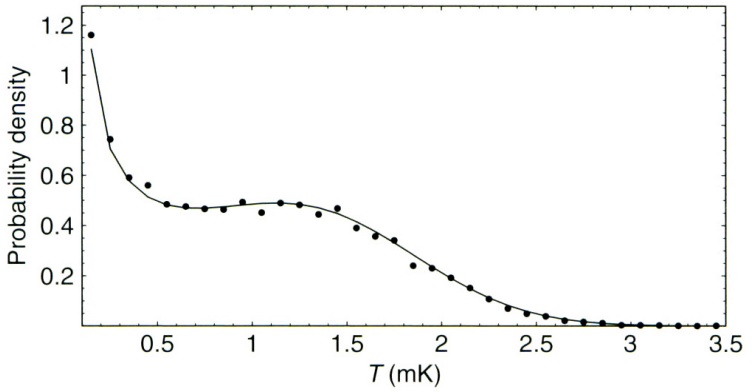
Conventional MCMC





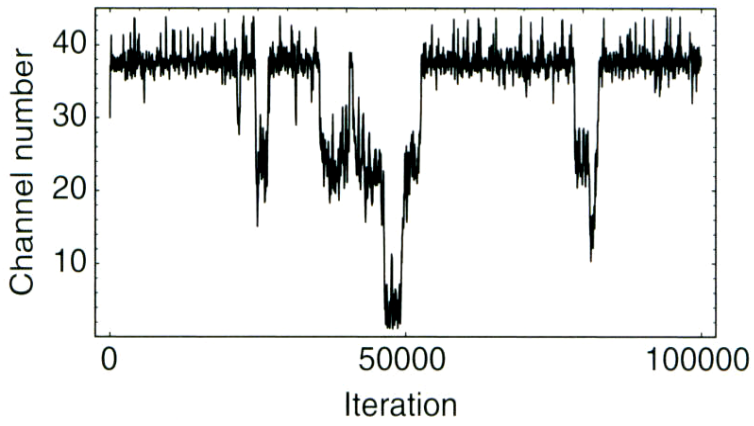
MCMC with parallel tempering

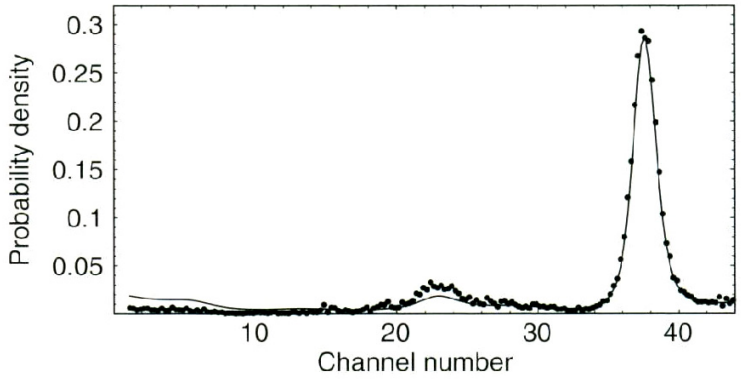




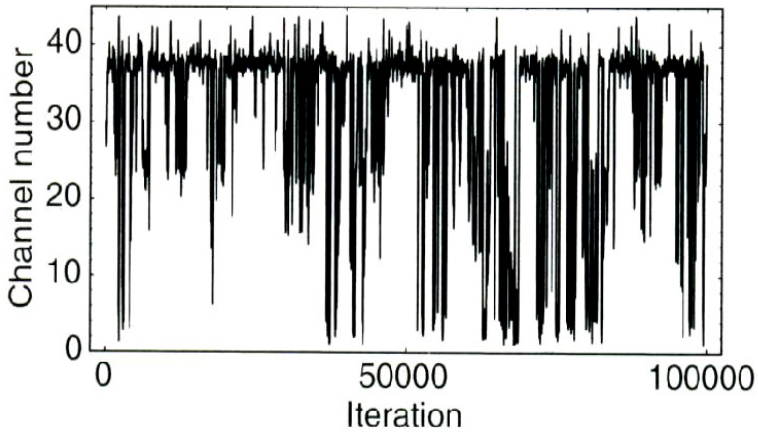
Example: spectral line fitting, from Section 3.

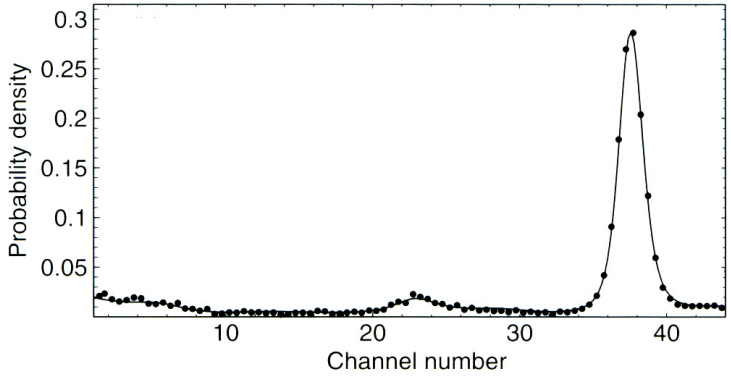
Conventional MCMC





MCMC with parallel tempering





Approximating the Evidence

$$\text{Evidence} = \int \mathrm{p}(\text{data}\,|\,\theta, M)\, p(\theta\,|\,M)\, d\theta$$

Average likelihood, weighted by prior

- Calculating the evidence can be computationally very costly (e.g. CMBR C_ℓ spectrum in cosmology)
- How to proceed?...

1. Information criteria (Liddle 2004, 2007)
2. Laplace and Savage-Dickey approximations (Trotta 2005)
3. Nested sampling (Skilling 2004, 2006; http://www.inference.phy.cam.ac.uk/bayesys/) (Mukherjee et al. 2005, 2007; Sivia 2006)

Nested Sampling (Skilling 2004, 2006; Mukherjee et al 2005, 2007)

$$\text{Evidence} = \int \text{p}(\text{data}\,|\,\theta, M)\, p(\theta\,|\,M)\, d\theta$$

Key idea:

We can rewrite the Evidence as

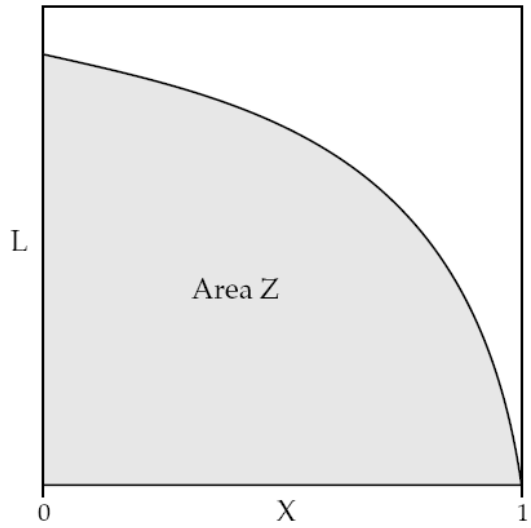
$$\text{Evidence} = \int \text{p}(\text{data}\,|\,\theta, M)\, \text{dX}$$

where X is a 1-D variable known as the ***prior mass*** uniformly distributed on [0,1]

$$\text{Evidence} = Z = \int_0^1 L(X)\, \text{dX}$$

Skilling (2006)

$$\text{Evidence} \quad = \quad Z \quad = \quad \int_0^1 L(X)\,\mathrm{dX}$$



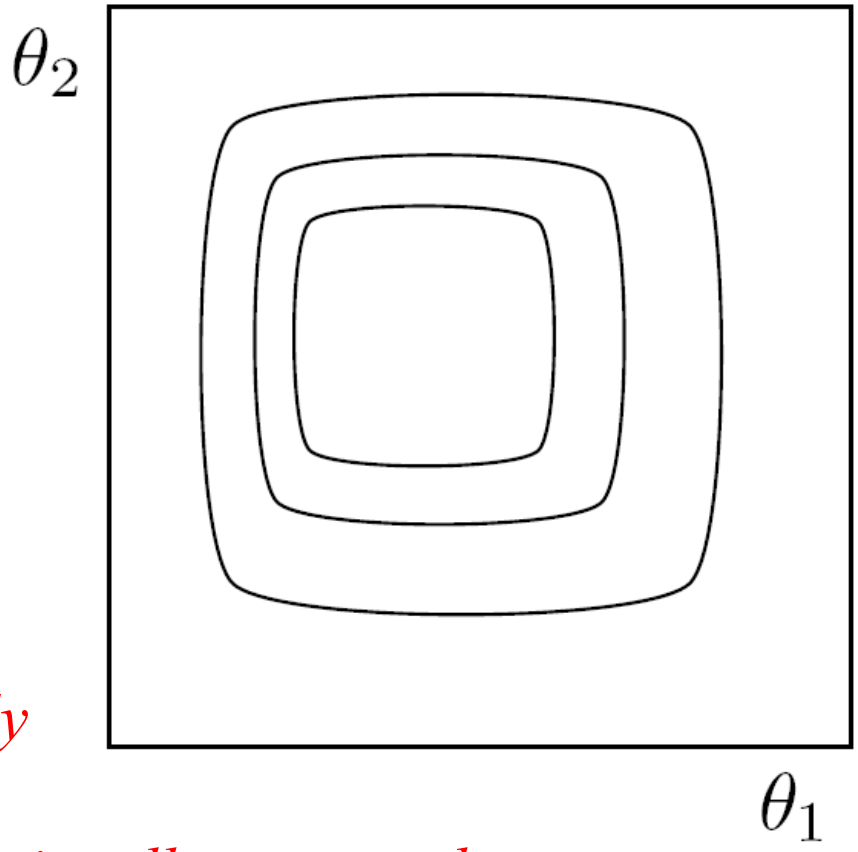
Example: 2-D Likelihood function

$$L \;=\; \begin{array}{|r|r|r|r|} \hline 0 & 8 & 15 & 3 \\ \hline 11 & 24 & 22 & 10 \\ \hline 19 & 30 & 26 & 16 \\ \hline 9 & 23 & 18 & 6 \\ \hline \end{array}$$

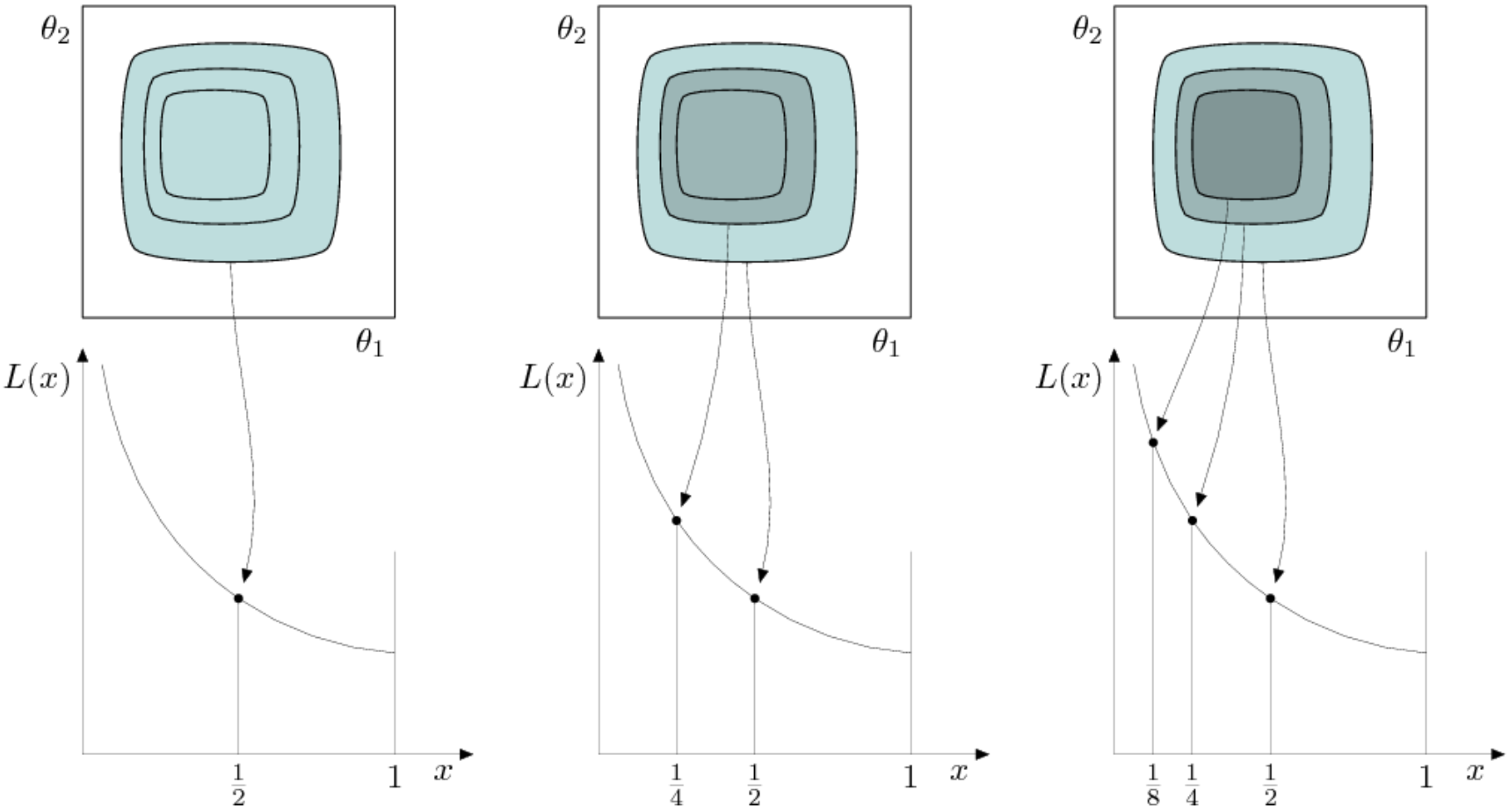
Our plan is to proceed as if we could sort these elements by likelihood, in the above example to $L = (30,26,24,23,22,19,18,16,15,11,10,9,8,6,3,0)$, whence $Z = \frac{30}{16}+\frac{26}{16}+\frac{24}{16}+\frac{23}{16}+\frac{22}{16}+\frac{19}{16}+\frac{18}{16}+\frac{16}{16}+\frac{15}{16}+\frac{11}{16}+\frac{10}{16}+\frac{9}{16}+\frac{8}{16}+\frac{6}{16}+\frac{3}{16}+\frac{0}{16} = 15$, to be evaluated right-to-left into domains of progressively greater likelihood. The likelihood corresponding to (say) $X = \frac{1}{5}$, being one fifth of the way along the sequence so falling into the fourth cell out of sixteen, is $L(X=0.2) = 23$.

Example: 2-D Likelihood function (from Mackay 2005)

- Contours of constant likelihood, L
- Each contour encloses a different fraction, X, of the area of the square
- Each point in the plane has an associated value of L and X



However, mapping systematically the relationship between L *and* X **everywhere** *may be computationally very costly*

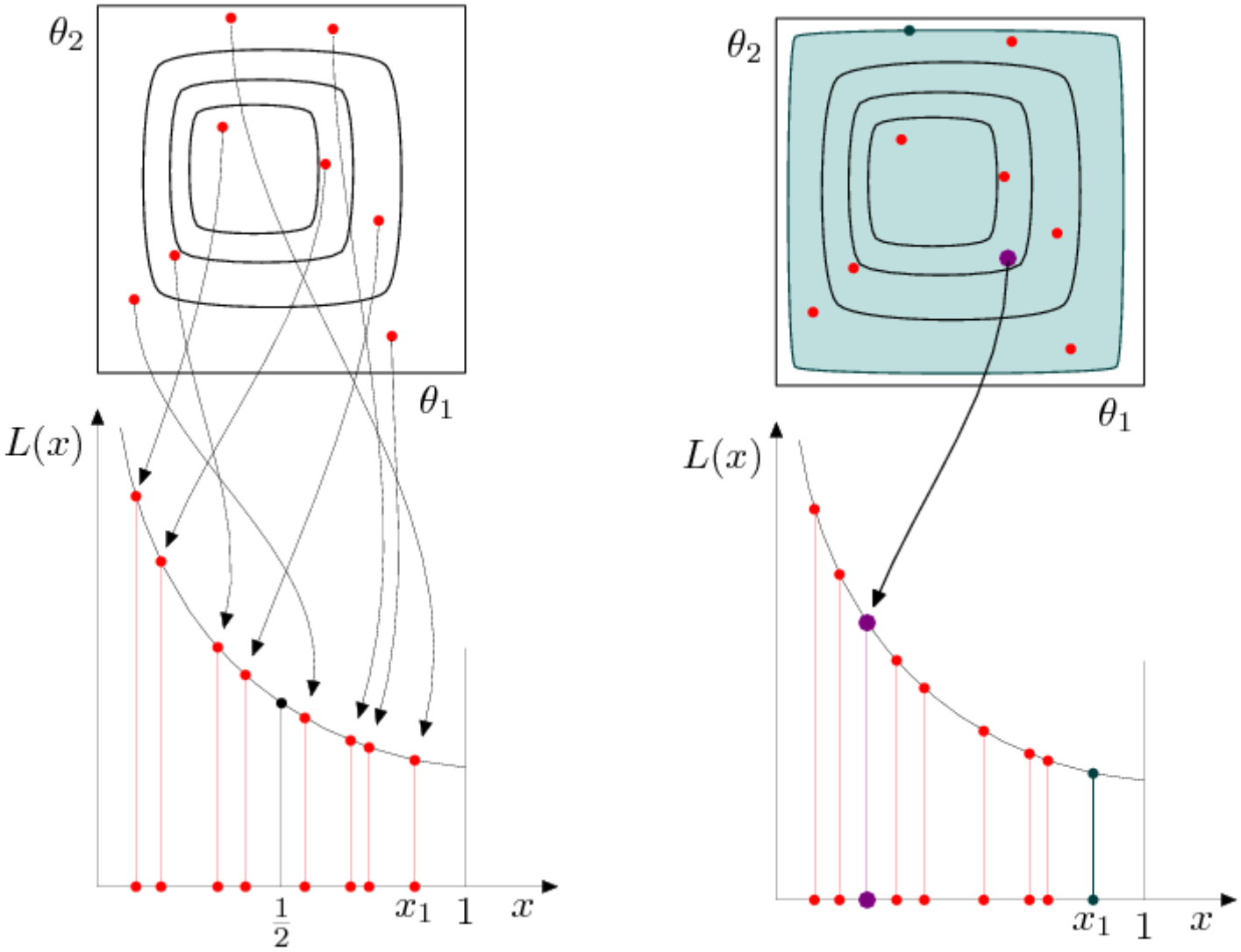


However, mapping systematically the relationship between
L *and* X **everywhere** *may be computationally very costly*

Skilling (2006)

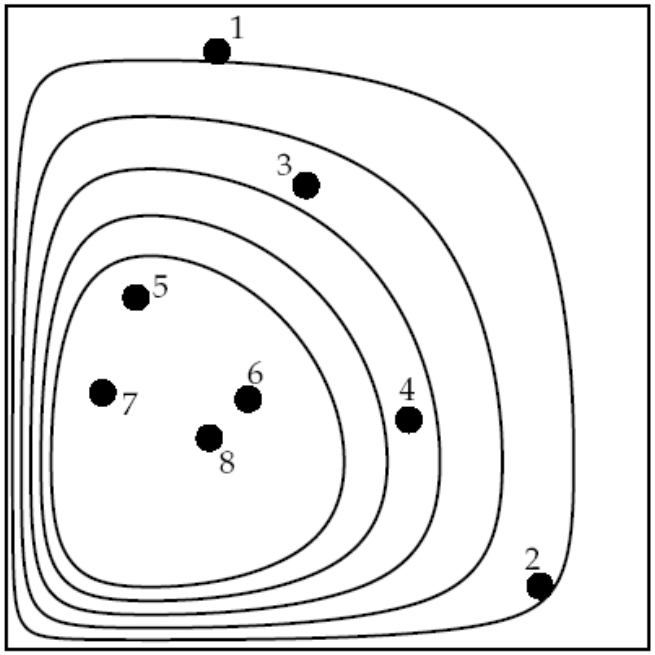
Approximation procedure

Start with N points $\theta_1, \ldots, \theta_N$ from prior;
initialise $Z = 0$, $X_0 = 1$.
Repeat for $i = 1, 2, \ldots, j$;
record the lowest of the current likelihood values as L_i,
set $X_i = \exp(-i/N)$ (crude) or sample it to get uncertainty,
set $w_i = X_{i-1} - X_i$ (simple) or $(X_{i-1} - X_{i+1})/2$ (trapezoidal),
increment Z by $L_i w_i$,
then replace point of lowest likelihood by new one drawn from within $L(\theta) > L_i$, in proportion to the prior $\pi(\theta)$.
Increment Z by $N^{-1}(L(\theta_1) + \ldots + L(\theta_N))\, X_j$.

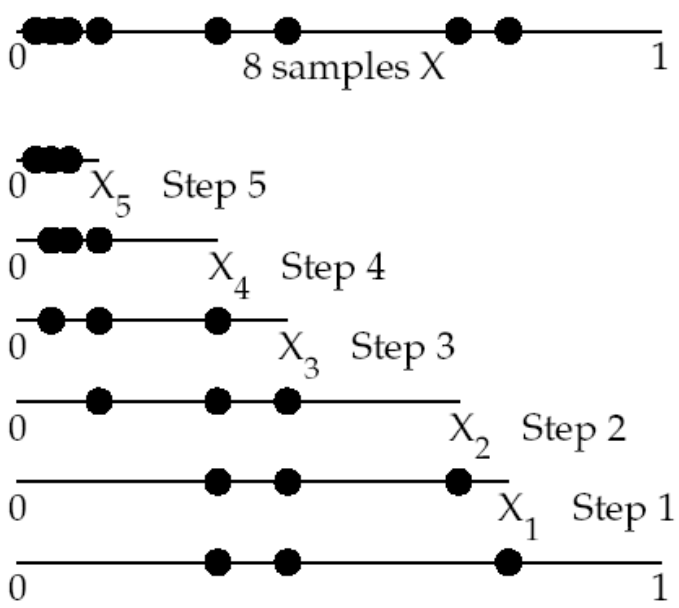


Let x_1 be the largest x-value. The typical value of x_1 is something like $N/(N+1)$ or $e^{-1/N}$. (The former is its arithmetic expected value, the latter its geometric mean.) We introduce a contour associated with this point.

Skilling (2006)



Parameter space



Enclosed prior mass X

NESTED SAMPLING TERMINATION

Termination of the main loop could simply be after a pre-set number of steps, or could be when even the largest current likelihood, taken over the full current box, would not increase the current evidence by more than some small fraction f;

$$\max(L_1, \ldots, L_N) X_j < f Z_j \implies \text{termination.} \qquad (16)$$

Plausibly, the accumulation of Z is then tailing off, so the sum is nearly complete.

7. Advanced Numerical Methods

Part 1: Monte Carlo Methods

Part 2: Fourier Methods

Part 2: Fourier Methods

In many diverse fields physical data is collected or analysed as Fourier components.

In this section we briefly discuss the mathematics of Fourier series and Fourier transforms.

1. Fourier Series

Any 'well-behaved' function $f(x)$ can be expanded in terms of an infinite sum of sines and cosines. The expansion takes the form:

$$f(x) = \tfrac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos nx + \sum_{n=1}^{\infty} b_n \sin nx$$

Joseph Fourier
(1768-1830)

The Fourier coefficients are given by the formulae:

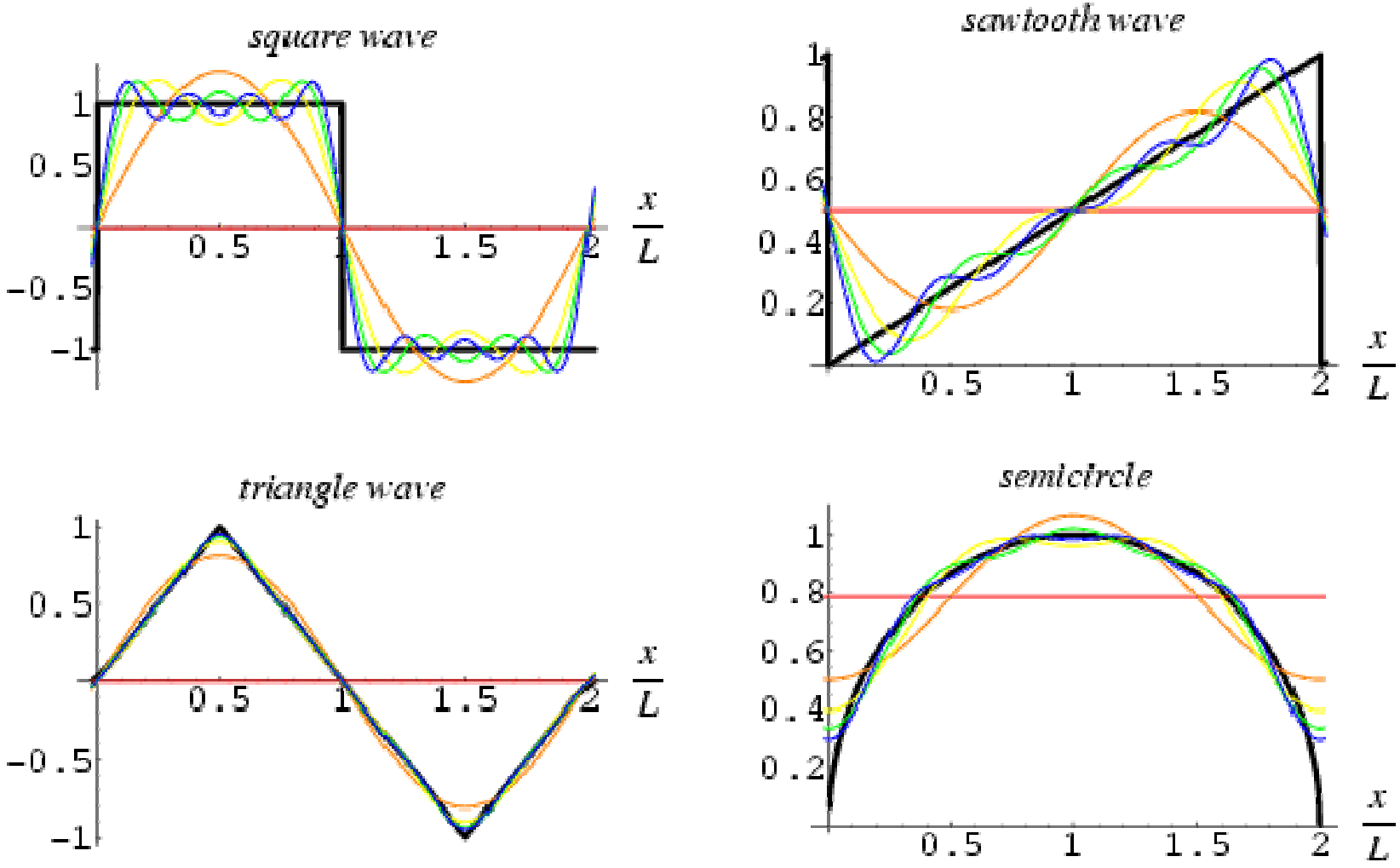
$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)dx$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos nxdx$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin nxdx$$

These formulae follow from the **orthogonality properties** of sin and cos:

$$\int_{-\pi}^{\pi} \sin mx \sin nxdx = \pi\delta_{mn} \qquad \int_{-\pi}^{\pi} \cos mx \cos nxdx = \pi\delta_{mn} \qquad \int_{-\pi}^{\pi} \sin mx \cos nxdx = 0$$



Some examples from Mathworld, approximating functions with a finite number of Fourier series terms

The Fourier series can also be written in complex form:

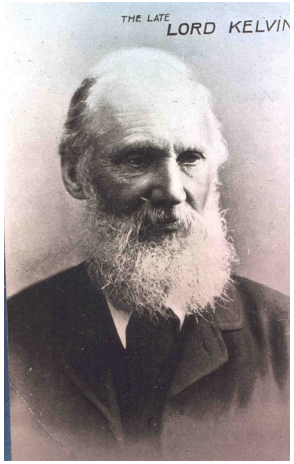
$$f(x) = \sum_{n=-\infty}^{\infty} A_n e^{inx}$$

where $$A_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-inx} dx$$

and recall that

$$e^{inx} = \cos nx + i \sin nx$$

$$e^{-inx} = \cos nx - i \sin nx$$



"Fourier's Theorem is not only one of the most beautiful results of modern analysis, but it is said to furnish an indispensable instrument in the treatment of nearly every recondite question in modern physics"

Fourier Transform: Basic Definition

The **Fourier transform** can be thought of simply as extending the idea of a Fourier series from an infinite sum over discrete, integer Fourier modes to an infinite integral over continuous Fourier modes.

Consider, for example, a physical process that is varying in the **time domain**, i.e. it is described by some function of time $h(t)$.

Alternatively we can describe the physical process in the **frequency domain** by defining the Fourier Transform function $H(f)$.

It is useful to think of $h(t)$ and $H(f)$ as two different representations of the same function; the information they convey about the underlying physical process should be equivalent.

We define the Fourier transform as

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt$$

and the corresponding inverse Fourier transform as

$$h(t) = \int_{-\infty}^{\infty} H(f) e^{-2\pi i f t} df$$

If time is measured in seconds then frequency is measured in cycles per second, or Hertz.

In many physical applications it is common to define the frequency domain behaviour of the function in terms of **angular frequency** $\omega = 2\pi f$

This changes the previous relations accordingly:

$$H(\omega) = \int_{-\infty}^{\infty} h(t) e^{2\pi i \omega t} dt$$

$$h(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} H(\omega) e^{-2\pi i \omega t} d\omega$$

Thus the symmetry of the previous expressions is broken.

Fourier Transform: Further properties

The FT is a **linear operation**:

(1) the FT of the sum of two functions is equal to the sum of their FTs
(2) the FT of a constant times a function is equal to the constant times the FT of the function.

If the time domain function $h(t)$ is a real function, then its FT is **complex**.

However, more generally we can consider the case where $h(t)$ is also a complex function – i.e. we can write

$$h(t) = h_R(t) + ih_I(t)$$

Real part

Imaginary part

$h(t)$ may also possess certain symmetries: **even function** $h(t) = h(-t)$

odd function $h(t) = -h(-t)$

The following properties then hold:

If . . .	then . . .
$h(t)$ is real	$H(-f) = [H(f)]^*$
$h(t)$ is imaginary	$H(-f) = -[H(f)]^*$
$h(t)$ is even	$H(-f) = H(f)$ [i.e., $H(f)$ is even]
$h(t)$ is odd	$H(-f) = -H(f)$ [i.e., $H(f)$ is odd]
$h(t)$ is real and even	$H(f)$ is real and even
$h(t)$ is real and odd	$H(f)$ is imaginary and odd
$h(t)$ is imaginary and even	$H(f)$ is imaginary and even
$h(t)$ is imaginary and odd	$H(f)$ is real and odd

See Numerical Recipes, Section 12.0

Note that in the above table a star (*) denotes the **complex conjugate**,

i.e. if $z = x + iy$ then $z^* = x - iy$

For convenience we will denote the FT pair by $h(t) \Leftrightarrow H(f)$

It is then straightforward to show that

$$h(at) \Leftrightarrow \frac{1}{|a|} H(f/a)$$ *"time scaling"*

$$\frac{1}{|b|} h(t/b) \Leftrightarrow H(bf)$$ *"frequency scaling"*

$$h(t - t_0) \Leftrightarrow H(f) e^{2\pi i f t_0}$$ *"time shifting"*

$$h(t)\, e^{-2\pi i f t_0} \Leftrightarrow H(f - f_0)$$ *"frequency scaling"*

Suppose we have two functions $g(t)$ and $h(t)$

Their **convolution** is defined as

$$(g * h)(t) = \int_{-\infty}^{\infty} g(s)h(t-s)\,ds$$

We can prove the **Convolution Theorem** $(g * h)(t) \Leftrightarrow G(f)H(f)$

i.e. the FT of the convolution of the two functions is equal to the product of their individual FTs.

Known as the ***lag***

Also their **correlation**, which is also a function of t, is defined as

$$\mathrm{Corr}(g,h) = \int_{-\infty}^{\infty} g(s+t)h(s)\,ds$$

We can prove the **Correlation Theorem** $\mathrm{Corr}(g,h) \Leftrightarrow G(f)H^*(f)$

i.e. the FT of the first time domain function, multiplied by the complex conjugate of the FT of the second time domain function, is equal to the FT of their correlation.

The correlation of a function with itself is called the **auto-correlation**

In this case $\mathrm{Corr}(g,g) \Leftrightarrow |G(f)|^2$

The function $|G(f)|^2$ is known as the **power spectral density**, or (more loosely) as the **power spectrum**.

Hence, the power spectrum is equal to the Fourier Transform of the auto-correlation function for the time domain function $g(t)$

The power spectral density

The power spectral density is analogous to the pdf we defined in previous sections.

In order to know how much power is contained in a given interval of frequency, we need to integrate the power spectral density over that interval.

The **total power** in a signal is the same, regardless of whether we measure it in the time domain or the frequency domain:

$$\text{Total Power} \equiv \int_{-\infty}^{\infty} |h(t)|^2 \, dt = \int_{-\infty}^{\infty} |H(f)|^2 \, df$$

Parseval's Theorem

Often we will want to know how much power is contained in a frequency interval without distinguishing between positive and negative values.

In this case we define the **one-sided power spectral density**:

$$P_h(f) \equiv |H(f)|^2 + |H(-f)|^2 \qquad 0 \le f < \infty$$

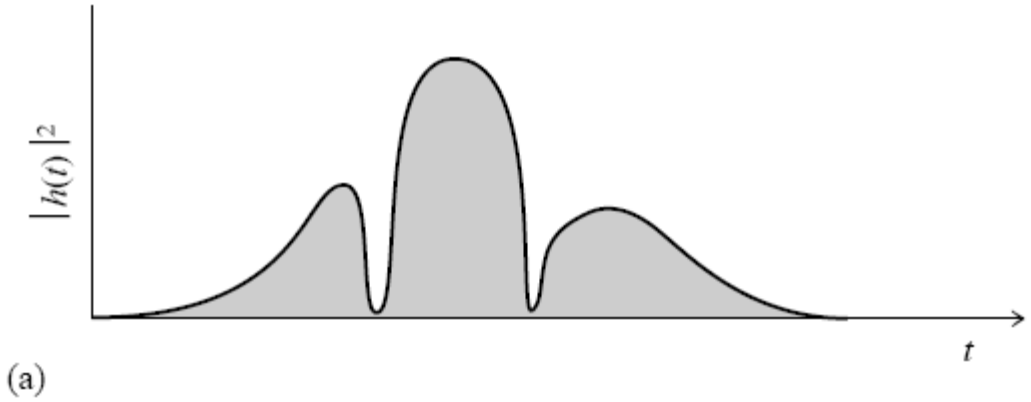
And

$$\text{Total Power} \equiv \int_0^\infty P_h(f)\,df$$

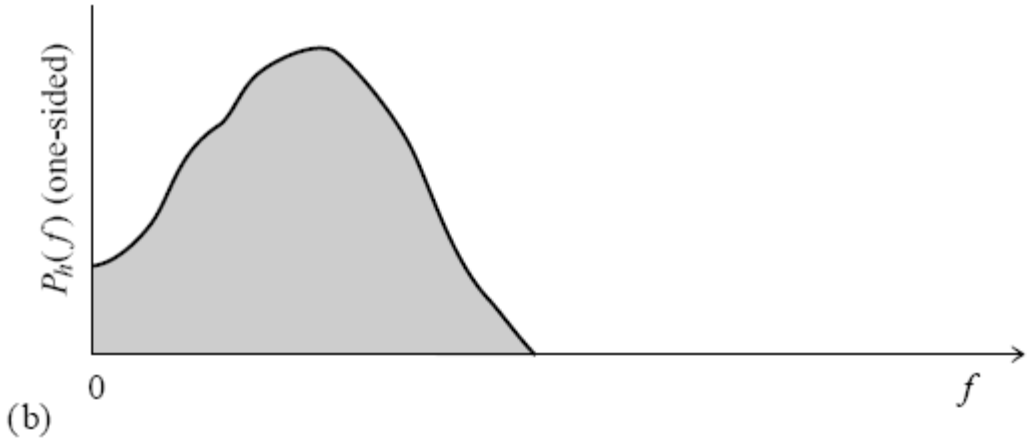
When $h(t)$ is a real function $P_h(f) \equiv 2|H(f)|^2$

With the proper normalisation, the total power (i.e. the integrated area under the relevant curve) is the same regardless of whether we are working with the time domain signal, the power spectral density or the one-sided power spectral density.

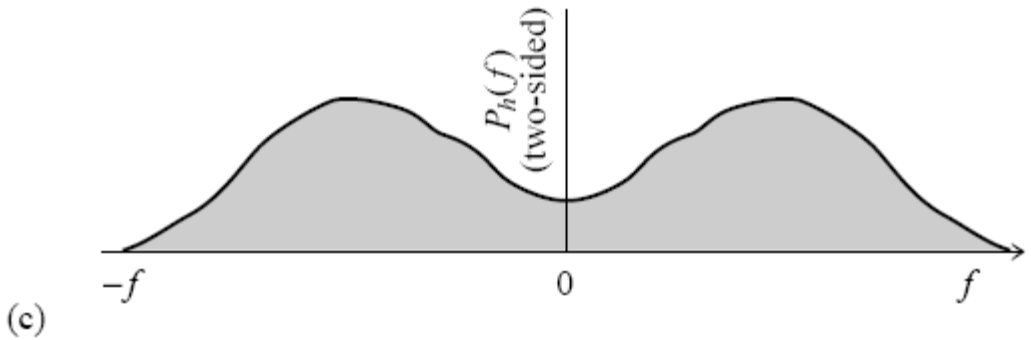
From Numerical Recipes, Chapter 12.0



Time domain



One-sided PSD

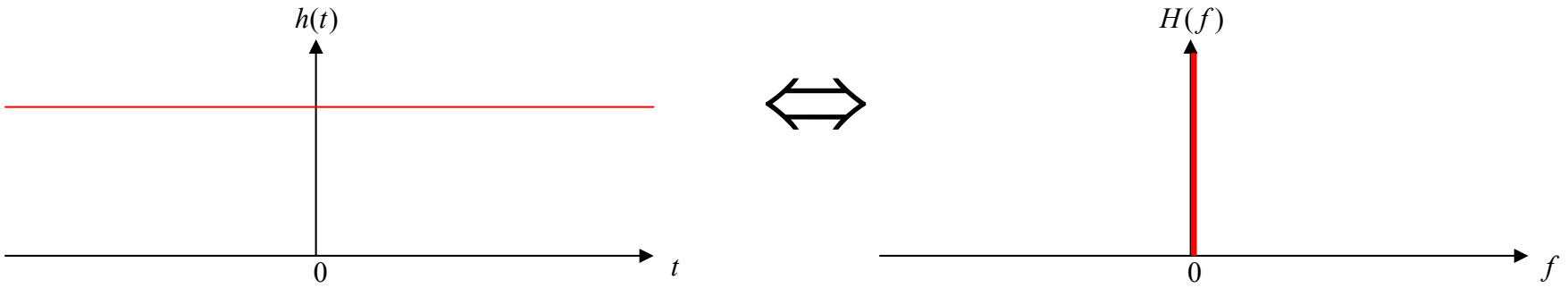


Two-sided PSD

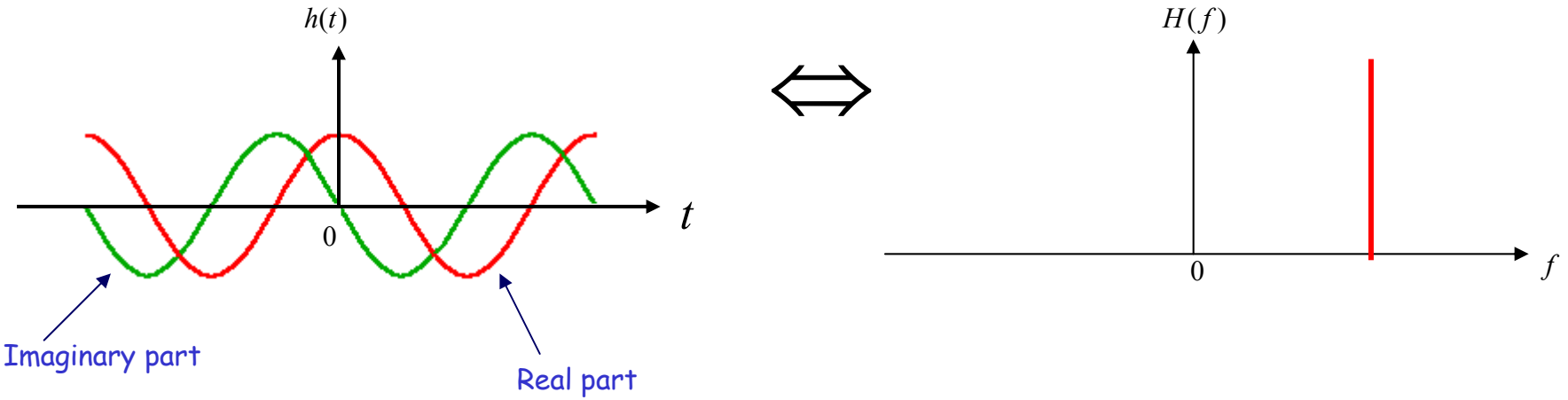
Examples

Dirac Delta function

(1) $h(t) = \text{const.}$ $\quad H(f) = \delta_{\mathrm{D}}(0)$

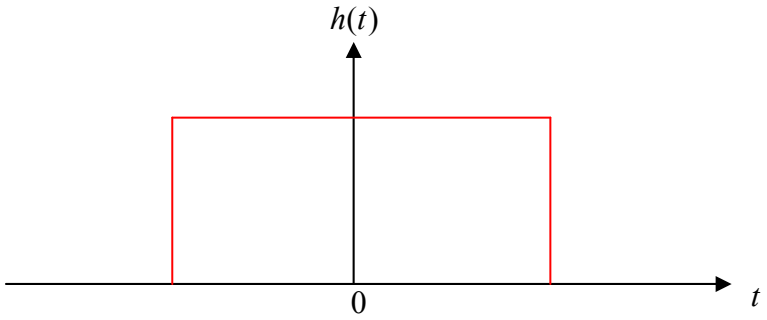


(2) $h(t) = e^{2\pi i f_0 t}$ $\quad H(f) = \delta_{\mathrm{D}}(f - f_0)$

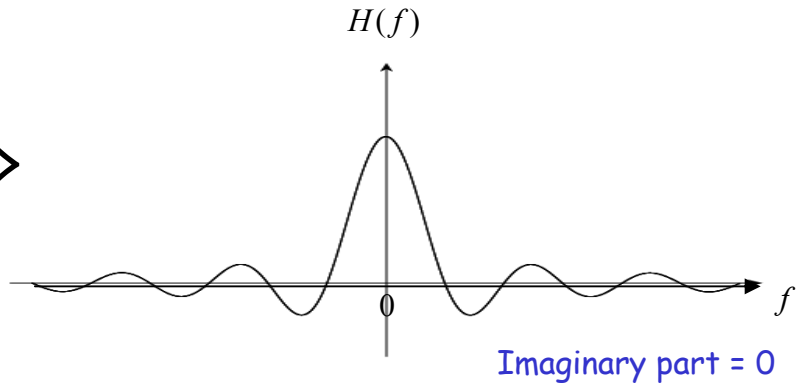


$$H(f) = \text{sinc}(\pi f)$$

(3) $$h(t) = \begin{cases} 1 & \text{for } -\frac{1}{2} \le t \le \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$



$\Leftrightarrow$

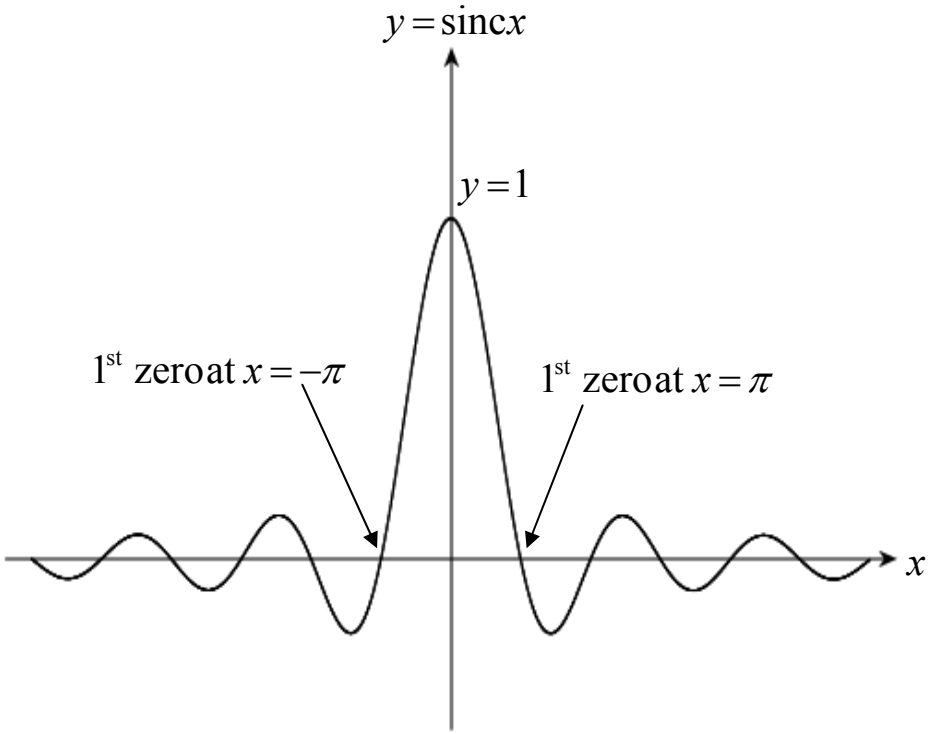


Imaginary part = 0

The sinc function occurs frequently in many areas of physics

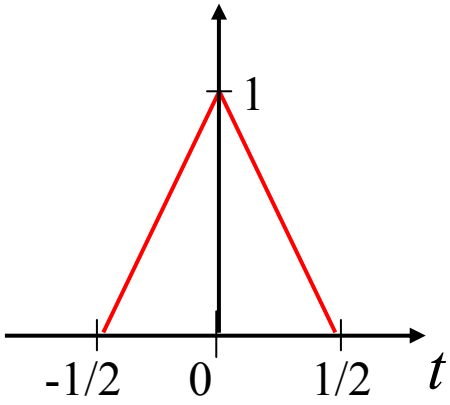
$$\text{sinc}\, x \equiv \frac{\sin x}{x}$$

The function has a maximum at $x = 0$ and the zeros occur at $x = \pm m\pi$ for positive integer m



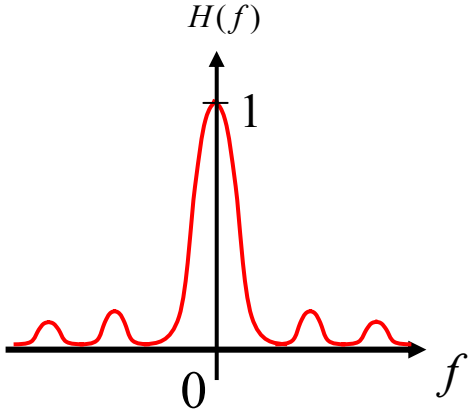
(4)

$$h(t) = \Delta(t)$$



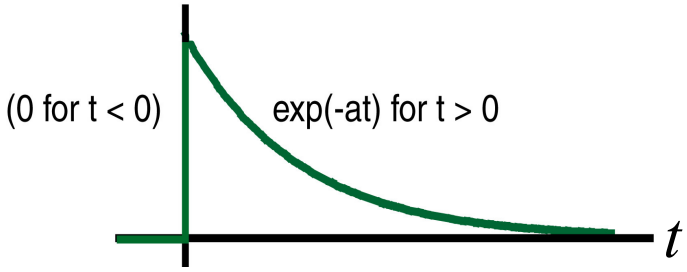
$\Leftrightarrow$

$$H(f) = \mathrm{sinc}^2(\pi f)$$



(5)

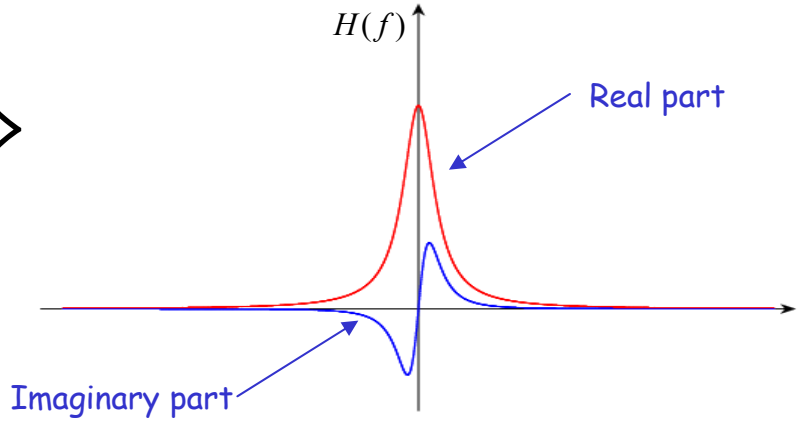
$$h(t) = e^{-at}$$



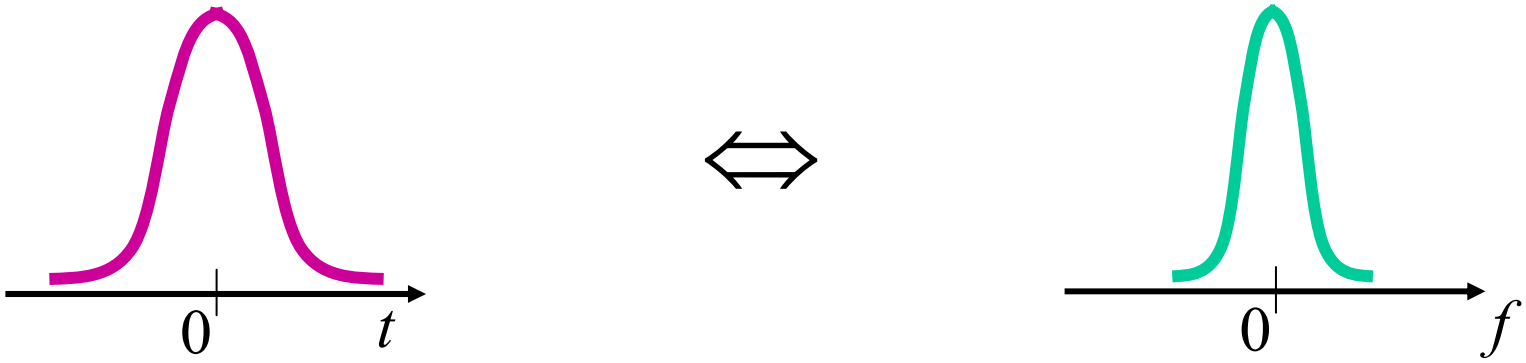
$\Leftrightarrow$

$$\mathrm{Re}\left[H(f)\right] = \frac{a^2}{a^2 + 4\pi^2 f^2}$$

$$\mathrm{Im}\left[H(f)\right] = -\frac{2a^2\pi f}{\left(a^2 + 4\pi^2 f^2\right)^2}$$



(6) $h(t) = \exp\left(-a^2 t^2\right)$ $H(f) \propto \exp\left(-\pi^2 f^2 / a^2\right)$



i.e. the FT of a Gaussian function in the time domain is *also* a Gaussian in the frequency domain.

Question 15: If the variance of a Gaussian is doubled in the time domain

A the variance of its Fourier transform will be doubled in the frequency domain

B the variance of its Fourier transform will be halved in the frequency domain

C the standard deviation of its Fourier transform will be doubled in the frequency domain

D the standard deviation of its Fourier transform will be halved in the frequency domain

Question 15: If the variance of a Gaussian is doubled in the time domain

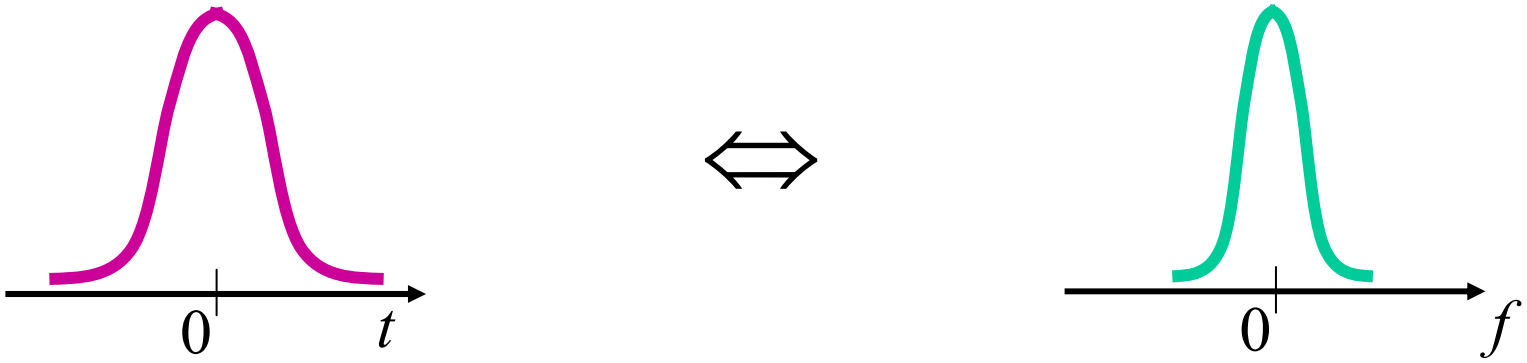
A the variance of its Fourier transform will be doubled in the frequency domain

B the variance of its Fourier transform will be halved in the frequency domain

C the standard deviation of its Fourier transform will be doubled in the frequency domain

D the standard deviation of its Fourier transform will be halved in the frequency domain

(6) $h(t) = \exp\left(-a^2 t^2\right)$ $H(f) \propto \exp\left(-\pi^2 f^2 / a^2\right)$



i.e. the FT of a Gaussian function in the time domain is *also* a Gaussian in the frequency domain.

The broader the Gaussian is in the time domain, then the narrower the Gaussian FT in the frequency domain, and vice versa.

Discrete Fourier Transforms

Although we have discussed FTs so far in the context of a continuous, analytic function, $h(t)$, in many practical situations we must work instead with observational data which are sampled at a discrete set of times.

Suppose that we sample $h(t)$ in total $N+1$ times at evenly spaced time intervals Δ, i.e. (for N even)

$$h_k \equiv h(t_k) \quad \text{where} \quad t_k = k\Delta, \qquad k = -N/2,...,0,...,N/2$$

[If $h(t)$ is non-zero over only a finite interval of time, then we suppose that the $N+1$ sampled points contain this interval. Or if $h(t)$ has an infinite range, then we at least suppose that the sampled points cover a sufficient range to be representative of the behaviour of $h(t)$].

We therefore approximate the FT as

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt \approx \sum_{k=-N/2}^{k=N/2} h_k e^{2\pi i f t_k} \Delta$$

Since we are sampling $h(t)$ at $N+1$ discrete timesteps, in view of the symmetry of the FT and inverse FT it makes sense also to compute $H(f)$ only at a set of $N+1$ discrete frequencies:

$$f_n \equiv \frac{n}{N\Delta}, \qquad n = -N/2, \ldots, 0, \ldots, N/2$$

(The frequency $f_c = 1/2\Delta$ is known as the **Nyquist (critical) frequency** and it is a very important value. We discuss its significance later).

Then

Discrete Fourier Transform of the h_k

$$H(f_n) \approx \sum_{k=-N/2}^{k=N/2} h_k e^{2\pi i f_n t_k} \Delta = \Delta \sum_{k=-N/2}^{k=N/2} h_k e^{2\pi i k n/N}$$

Note that $e^{-\pi i n} = e^{\pi i n}$

Hence, there are only N independent values.

Also, note that $e^{2\pi i k n/N} = e^{2\pi i n + 2\pi i k n/N} = e^{2\pi i n(N+k)/N}$

So we can re-define the Discrete FT as:

$$H(f_n) \approx \Delta \sum_{k=0}^{N-1} h_k e^{2\pi i k n/N} = \Delta H_n$$

The discrete *inverse* FT, which recovers the set of h_k's from the set of H_n's is

$$h_k = \frac{1}{N}\sum_{k=0}^{N-1} H_n e^{-2\pi i k n/N}$$

Parseval's theorem for discrete FTs takes the form

$$\sum_{k=0}^{N-1}|h_k|^2 \quad = \quad \frac{1}{N}\sum_{n=0}^{N-1}|H_n|^2$$

There are also discrete analogues to the convolution and correlation theorems.

Fast Fourier Transforms

Consider again the formula for the discrete FT. We can write it as

$$H_n = \sum_{k=0}^{N-1} e^{2\pi i k n/N} h_k \equiv \sum_{k=0}^{N-1} W^{nk} h_k$$

This is a **matrix equation**: we compute the $(N \times 1)$ vector of H_n's by multiplying the $(N \times N)$ matrix $\left[W^{nk}\right]$ by the $(N \times 1)$ vector of h_k's.

In general, this requires of order N^2 multiplications (and the h_k's may be complex numbers).

e.g. suppose $N = 10^8 \Rightarrow N^2 = 10^{16}$. Even if a computer can perform (say) 1 billion multiplications per second, it would still require more than **115 days** to calculate the FT.

Fortunately, there is a way around this problem.

Suppose (as we assumed before) N is an even number. Then we can write

$$H_n = \sum_{k=0}^{N-1} e^{2\pi i k n/N} h_k = \underbrace{\sum_{j=0}^{N/2-1} e^{2\pi i (2j) n/N} h_{2j}}_{\text{Even values of } k} + \underbrace{\sum_{j=0}^{N/2-1} e^{2\pi i (2j+1) n/N} h_{2j+1}}_{\text{Odd values of } k}$$

$$= \sum_{j=0}^{M-1} e^{2\pi i j n/M} h_{2j} + W^n \sum_{j=0}^{M-1} e^{2\pi i j n/M} h_{2j+1}$$

where $M = N/2$

So we have turned an FT with N points into the weighted sum of **two** FTs with $N/2$ points. This would reduce our computing time by a factor of two.

Why stop there, however?...

If M is *also* even, we can repeat the process and re-write the FTs of length M as the weighted sum of two FTs of length $M/2$.

$$\vdots$$

If N is a **power of two** (e.g. 1024, 2048, 1048576 etc) then we can repeat iteratively the process of splitting each longer FT into two FTs half as long.

The final step in this iteration consists of computing FTs of length unity:

$$H_0 = \sum_{k=0} e^{2\pi i k 0} h_k \equiv h_0$$

i.e. the FT of each discretely sampled data value is just the data value itself.

This iterative process converts $O(N^2)$ multiplications into $O(N \log_2 N)$ operations.

This notation means 'of the order of'

So our 10^{16} operations are reduced to about 2.7×10^9 operations.

Instead of 100 days of CPU time, we can perform the FT in less than 3 seconds.

The Fast Fourier Transform (FFT) has revolutionised our ability to tackle problems in Fourier analysis on a desktop PC which would otherwise be impractical, even on the largest supercomputers.

Data Acquisition

Earlier we approximated the continuous function $h(t)$ and its FT $H(f)$ by a finite set of $N+1$ discretely sampled values.

How good is this approximation? The answer depends on the form of $h(t)$ and $H(f)$. In this short section we will consider:

1. under what conditions we can reconstruct $h(t)$ and $H(f)$ ***exactly*** from a set of discretely sampled points?

2. what is the minimum **sampling rate** (or density, if h is a spatially varying function) required to achieve this exact reconstruction?

3. what is the effect on our reconstructed $h(t)$ and $H(f)$ if our data acquisition does *not* achieve this minimum sampling rate?

The Nyquist – Shannon Sampling Theorem

Suppose the function $h(t)$ is **bandwidth limited**. This means that the FT of $h(t)$ is non-zero over a finite range of frequencies.

i.e. there exists a **critical frequency** f_c such that

$$H(f) = 0 \quad \text{for all} \quad |f| \geq f_c$$

The **Nyquist – Shannon Sampling Theorem** (NSST) is a very important result from information theory. It concerns the representation of $h(t)$ by a set of discretely sampled values

$$h_k \equiv h(t_k) \quad \text{where} \quad t_k = k\Delta, \qquad k = \ldots,-2,-1,0,1,2,\ldots$$

The NSST states that, provided the sampling interval Δ satisfies

$$\Delta = 1/2f_c$$ or less

then we can **exactly** reconstruct the function $h(t)$ from the discrete samples $\{h_k\}$. It can be shown that

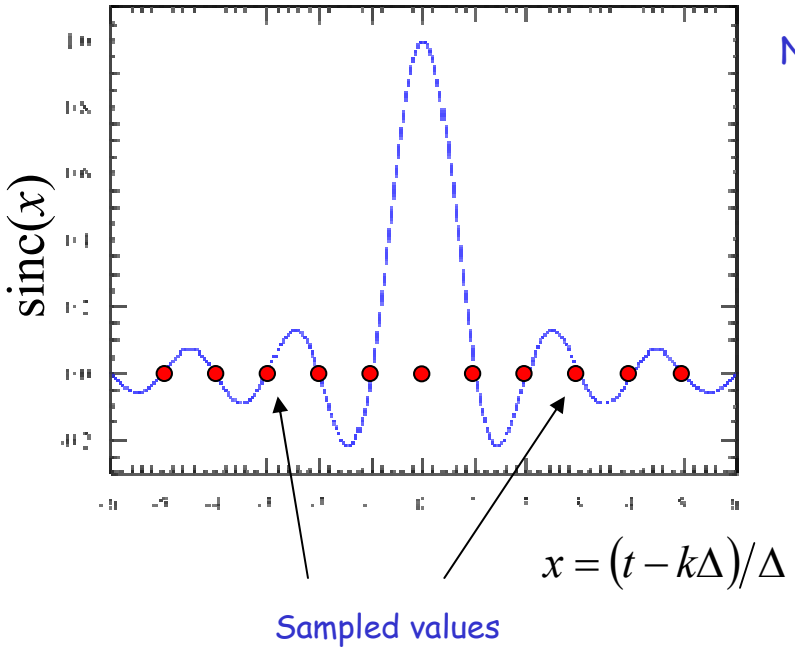
$$h(t) = \Delta \sum_{k=-\infty}^{+\infty} h_k \frac{\sin[2\pi f_c(t - k\Delta)]}{\pi(t - k\Delta)}$$

f_c is also known as the **Nyquist frequency** and $\Delta^{-1} = 2f_c$ is known as the **Nyquist rate**.

We can re-write this as

$$h(t) = \sum_{k=-\infty}^{+\infty} h_k \frac{\sin\left[\pi\left(t - k\Delta\right)/\Delta\right]}{\pi\left[\left(t - k\Delta\right)/\Delta\right]}$$

So the function $h(t)$ is the sum of the sampled values $\{h_k\}$, weighted by the ***normalised*** **sinc function**, scaled so that its zeroes lie at those sampled values.



Normalised sinc function

$$\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

(compare with previous section)

Note that when $t = k\Delta$ then $h(t) = h_k$ since $\mathrm{sinc}(0) = 1$

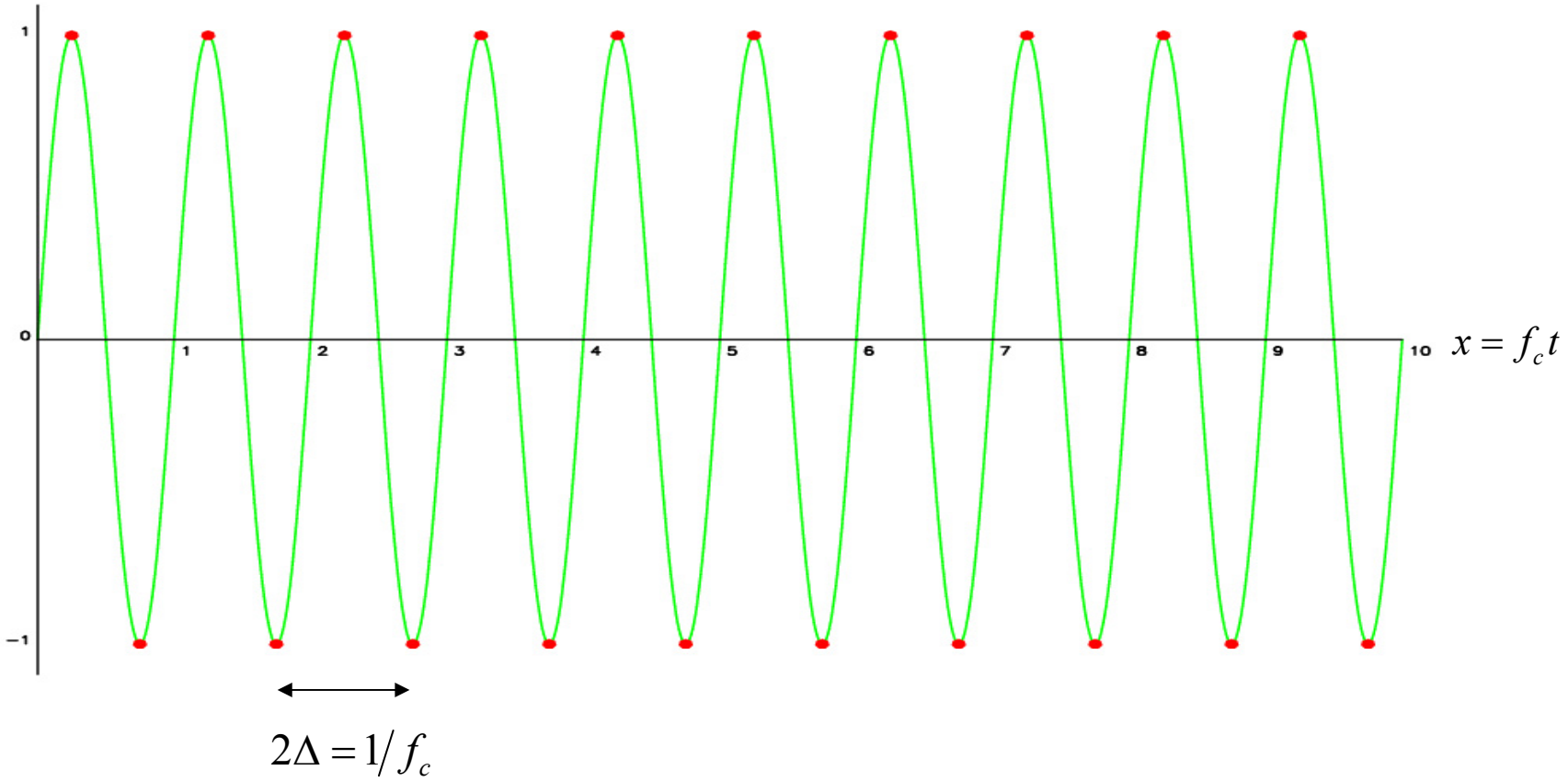
The NSST is a very powerful result.

We can think of the interpolating sinc functions, centred on each sampled point, as 'filling in the gaps' in our data. The remarkable fact is that they do this job *perfectly*, provided $h(t)$ is bandwidth limited. i.e. the discrete sampling incurs no loss of information about $h(t)$ and $H(f)$.

Suppose, for example, that $h(t) = \sin(2\pi f_c t)$. Then we need only sample $h(t)$ *twice* every period in order to be able to reconstruct the entire function exactly.

(Note that formally we do need to sample an *infinite number* of discretely spaced values, $\{h_k\}$. If we only sample the $\{h_k\}$ over a finite time interval, then our interpolated $h(t)$ will be approximate).

$y = \sin(2\pi f_c t)$



Sampling $h(t)$ at (infinitely many of) the **red** points is sufficient to reconstruct the function for all values of t, with no loss of information.

Aliasing

There is a downside, however.

If $h(t)$ is *not* bandwidth limited (or, equivalently, if we don't sample frequently enough – i.e. if the sampling rate $\Delta^{-1} < 2f_c$) then our reconstruction of $h(t)$ and $H(f)$ is badly affected by **aliasing**.

This means that all of the power spectral density which lies *outside* the range $-f_c < f < f_c$ is spuriously moved *inside* that range, so that the FT $H(f)$ of $h(t)$ will be computed **incorrectly** from the discretely sampled data.

Any frequency component outside the range $(-f_c, f_c)$ is falsely translated (**aliased**) into that range.

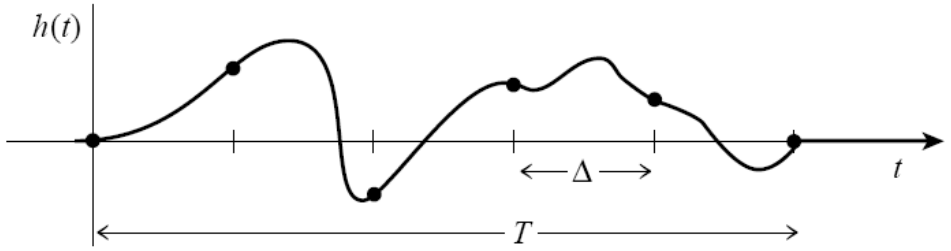
Consider $h(t)$ as shown.

Suppose $h(t)$ is zero outside the range T.

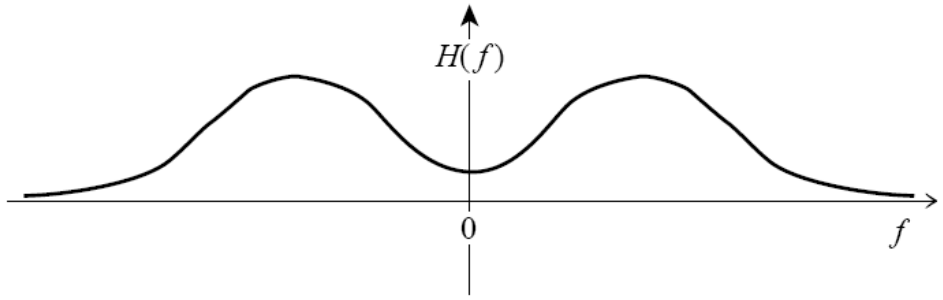
This means that $H(f)$ extends to $\pm\infty$.

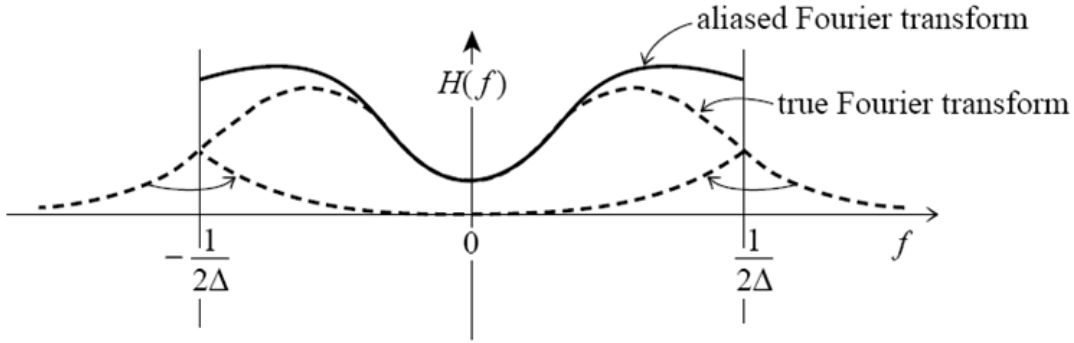
The contribution to the true FT from outside the range $(-1/2\Delta, 1/2\Delta)$ gets aliased into this range, appearing as a 'mirror image'.

Thus, at $f = \pm 1/2\Delta$ our computed value of $H(f)$ is equal to *twice* the true value.



$h(t)$ sampled at regular intervals Δ





From Numerical Recipes, Chapter 12.1

How do we combat aliasing?

- Enforce some chosen f_c e.g. by *filtering* $h(t)$ to remove the high frequency components $|f| > f_c$. *(Also known as anti-aliasing)*

- Sample $h(t)$ at a high enough rate Δ^{-1} so that $\Delta^{-1} \geq 2f_c$ - i.e. at least two samples per cycle of the highest frequency present

To check for / eliminate aliasing *without* pre-filtering:

- Given a sampling interval Δ, compute $f_{\text{lim}} = 1/2\Delta$

- Check if discrete FT of $h(t)$ is approaching **zero** as $f \to f_{\text{lim}}$

- If *not*, then frequencies outside the range $(-1/2\Delta, 1/2\Delta)$ are probably being folded back into this range.

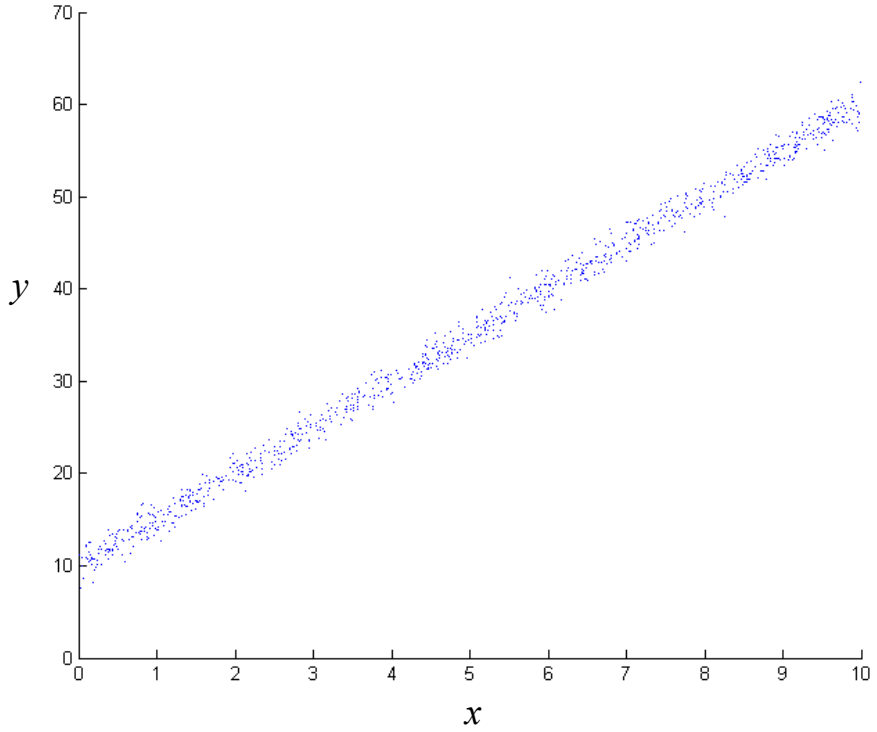
- Try increasing the sampling rate, and repeat…

And finally….

Mock Data Challenge

1000 (x,y) data pairs, generated from an unknown model plus Gaussian noise.

Data posted on my.SUPA



Four-stage challenge:

1. Fit a linear model to these data, using ordinary least squares;
2. Compute Bayesian credible regions for the model parameters, using a bivariate normal model for the likelihood function;
3. Write an MCMC code to sample from the posterior pdf of the model parameters, and compare their sample estimates with the LS fits;
4. Carry out a Bayesian model comparison, calculating the posterior odds ratio of a constant, linear and quadratic model.