Numerical Algorithms for HPC

Monte Carlo Methods

Introduction to Monte Carlo methods

Overview

- Integration by random numbers
  - why? how?
  - accuracy?
- Algorithms
  - importance sampling
  - Markov Chain Monte Carlo
  - optimisation
- Examples
  - statistical physics
  - finance
  - weather forecasting

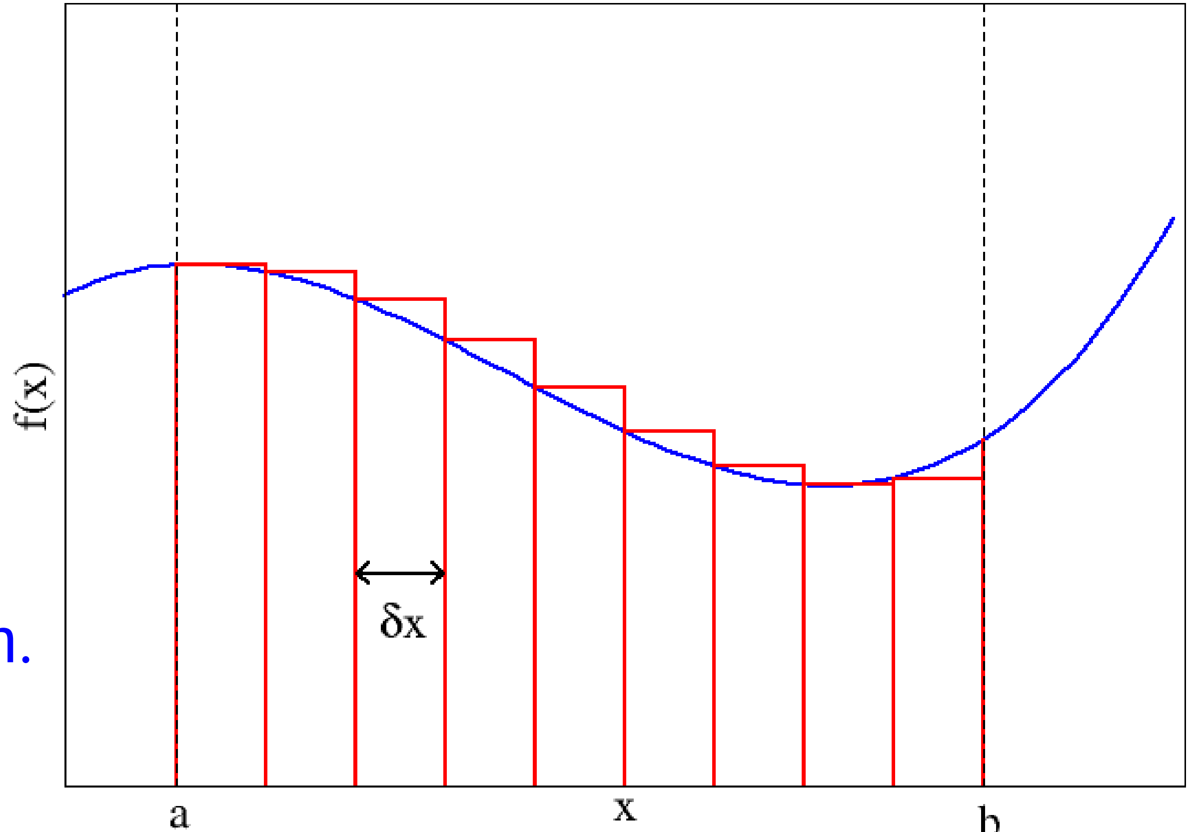
Integration – Area under a curve

Tile area with strips of height *f(x)* and width *δx*

Analytical:

$$\delta x \rightarrow dx \rightarrow 0$$

Numerical: integral replaced with a sum.

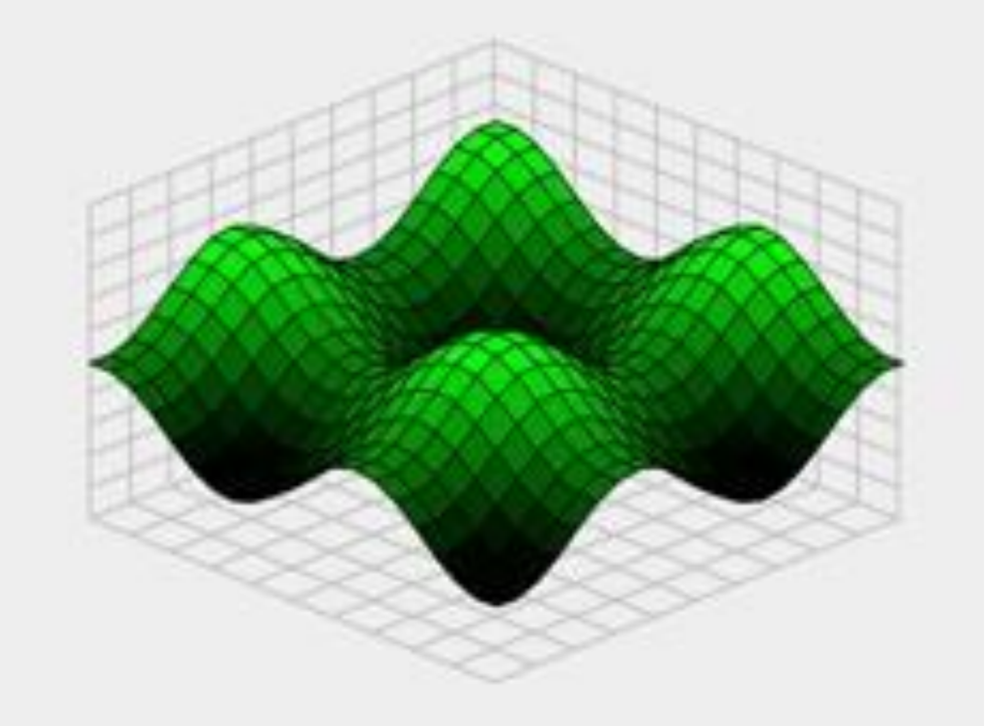


Uncertainty depends on size of *δx* (*N* points) and order of scheme, (Trapezoidal, Simpson, etc)

Multi-dimensional integration

1d integration requires N points

2d integration requires N^2



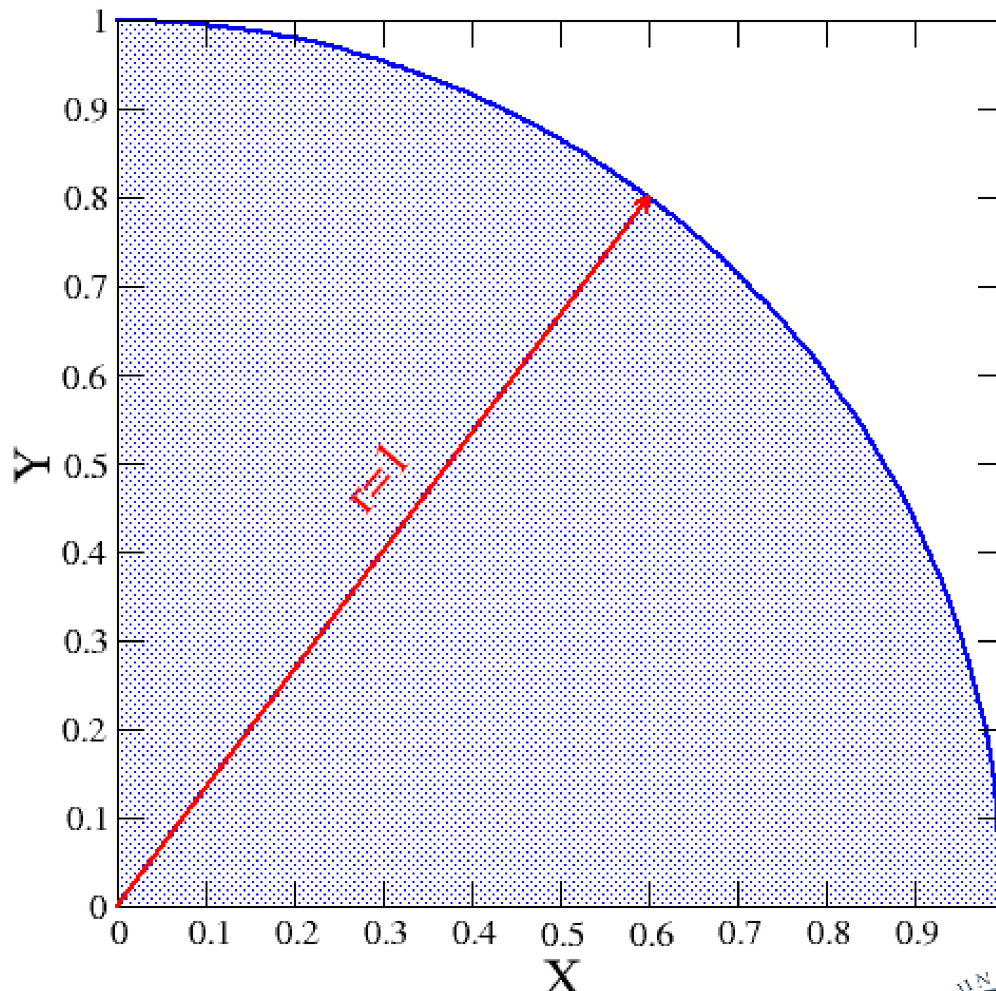
Problem of dimension m requires N^m

The "Curse of dimensionality"

Calculating π by Monte Carlo (MC)

Area of circle = πr^2
Area of unit square, $s = 1$
Area of shaded arc,
$c = \pi/4$
$c/s = \pi/4$

Estimate ratio of shaded to non-shaded area to determine π



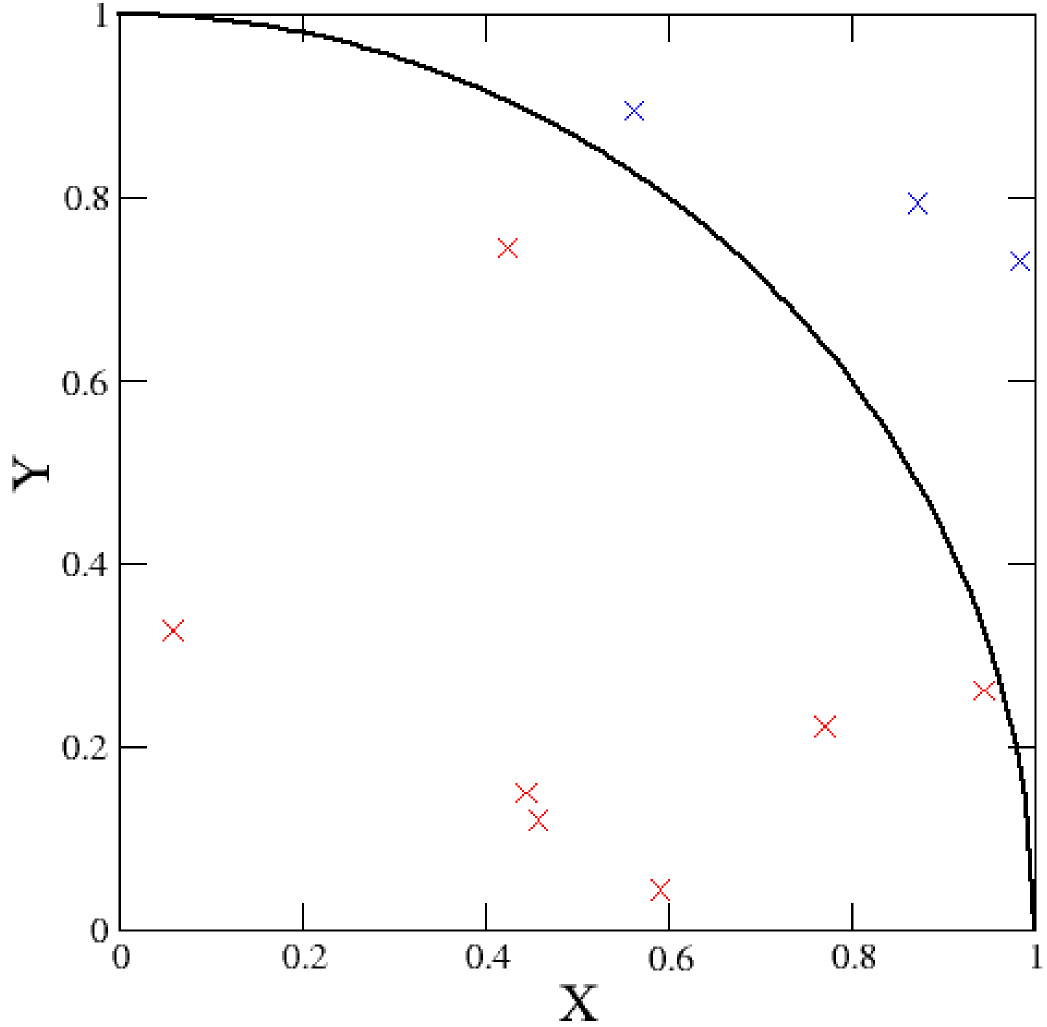
epcc

The algorithm

```
y = random()  // float [0.0:1.0)
x = random()
p = x*x + y*y //x*x + y*y = 1, eqn of circle
If (p <= 1)
  isInCircle++
Else
  IsOutCircle++
Pi=4*isInCircle / (isOutCircle+isInCircle)
```

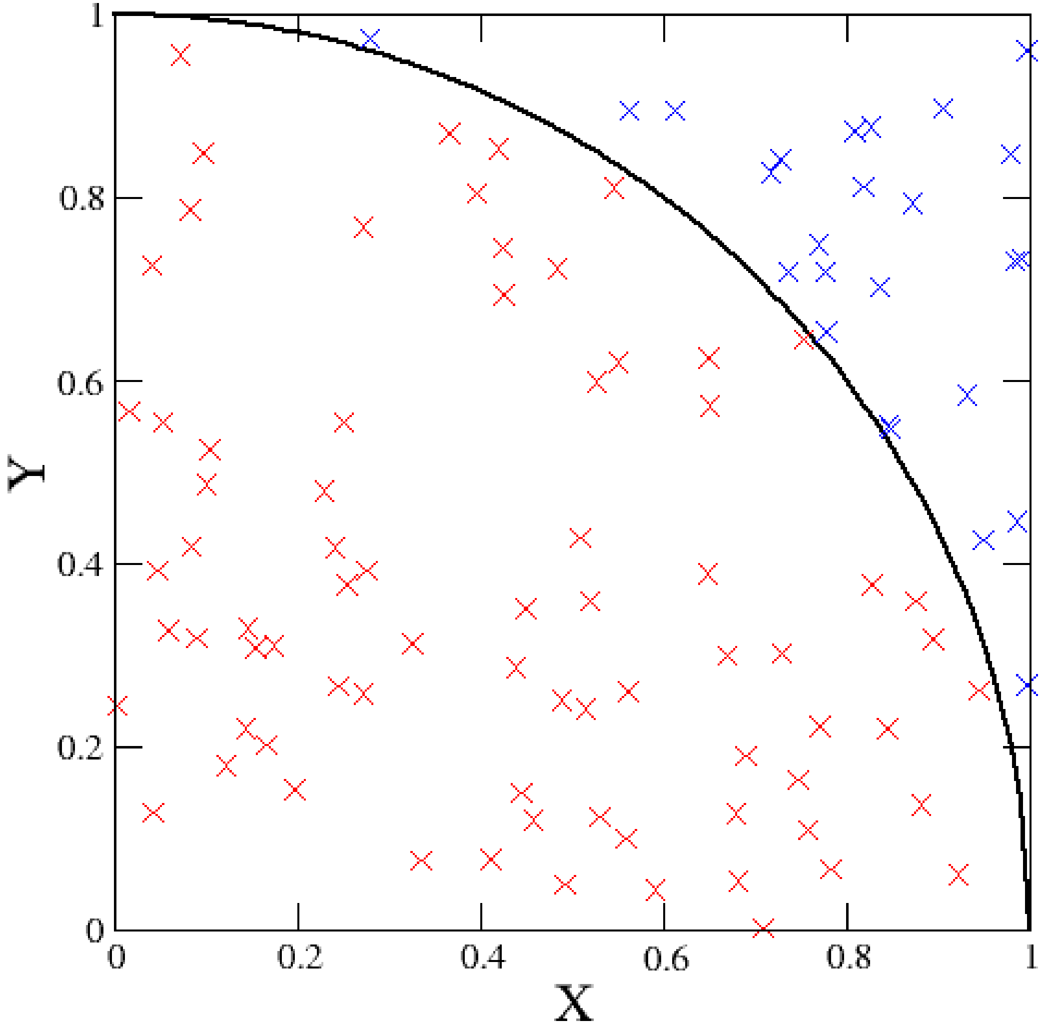
π from 10 darts

$\pi = 2.8$



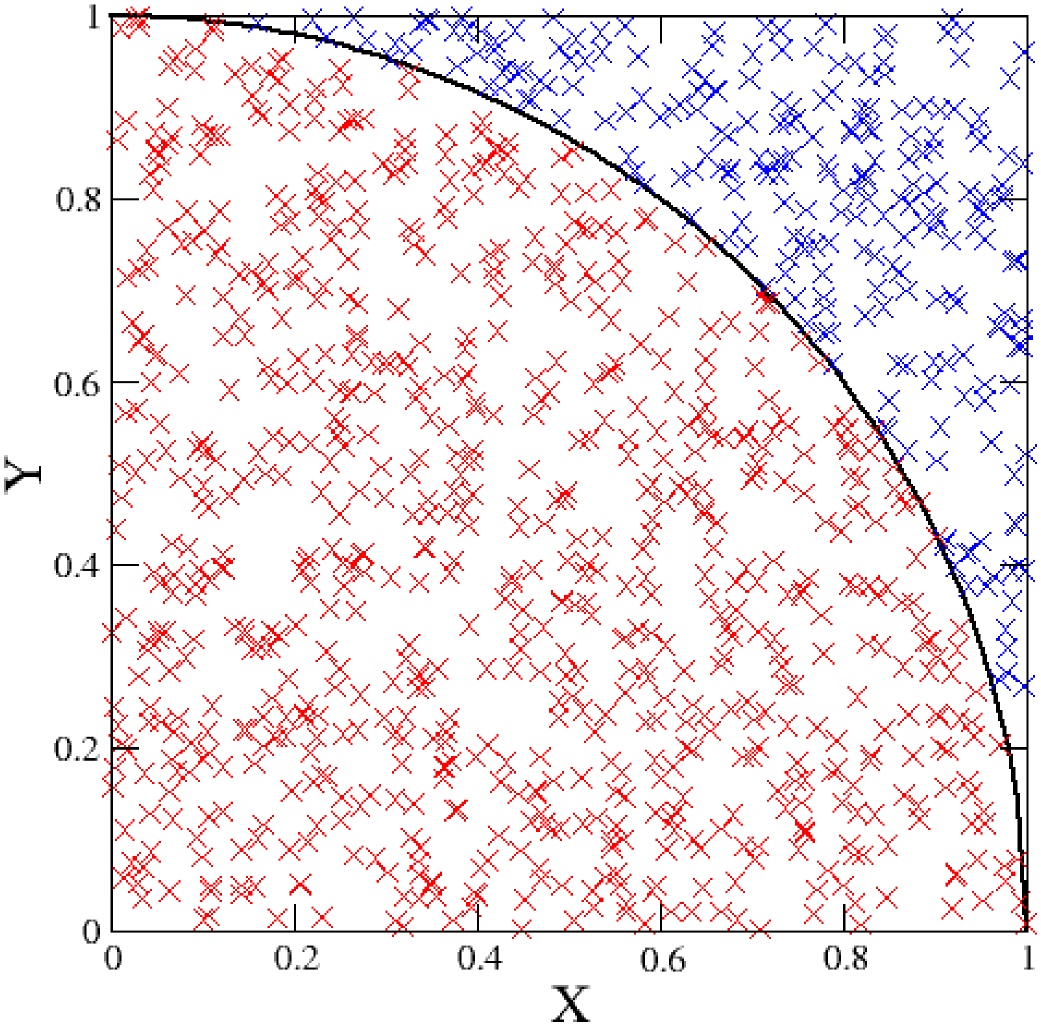
π from 100 darts

$\pi = 3.0$



π from 1000 darts

$\pi = 3.12$

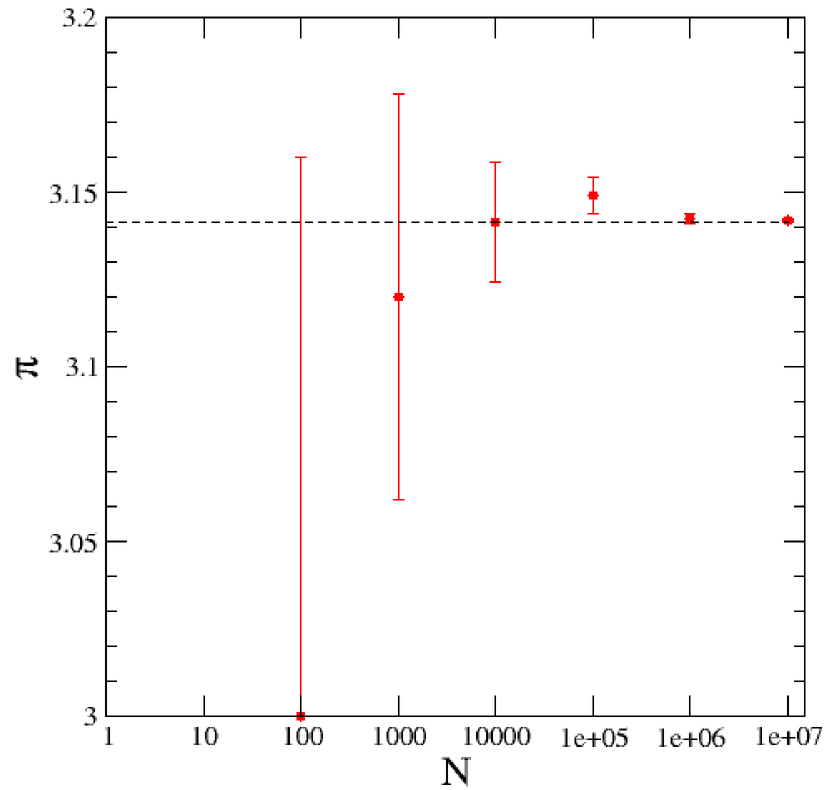


Estimating the uncertainty

- A random or ***stochastic*** method
  - leads to statistical uncertainty
- Estimate this
  - run each measurement *k* times with *different random number sequences*
  - determine the variance of the distribution (plot has *k* = 100)

$$\sigma^2 = \sum_{i=1}^{k}(x_i - \bar{x})^2/(k\text{-}1)$$

- Standard deviation is σ
  - how does the uncertainty scale with *N*, number of samples?



Uncertainty versus N

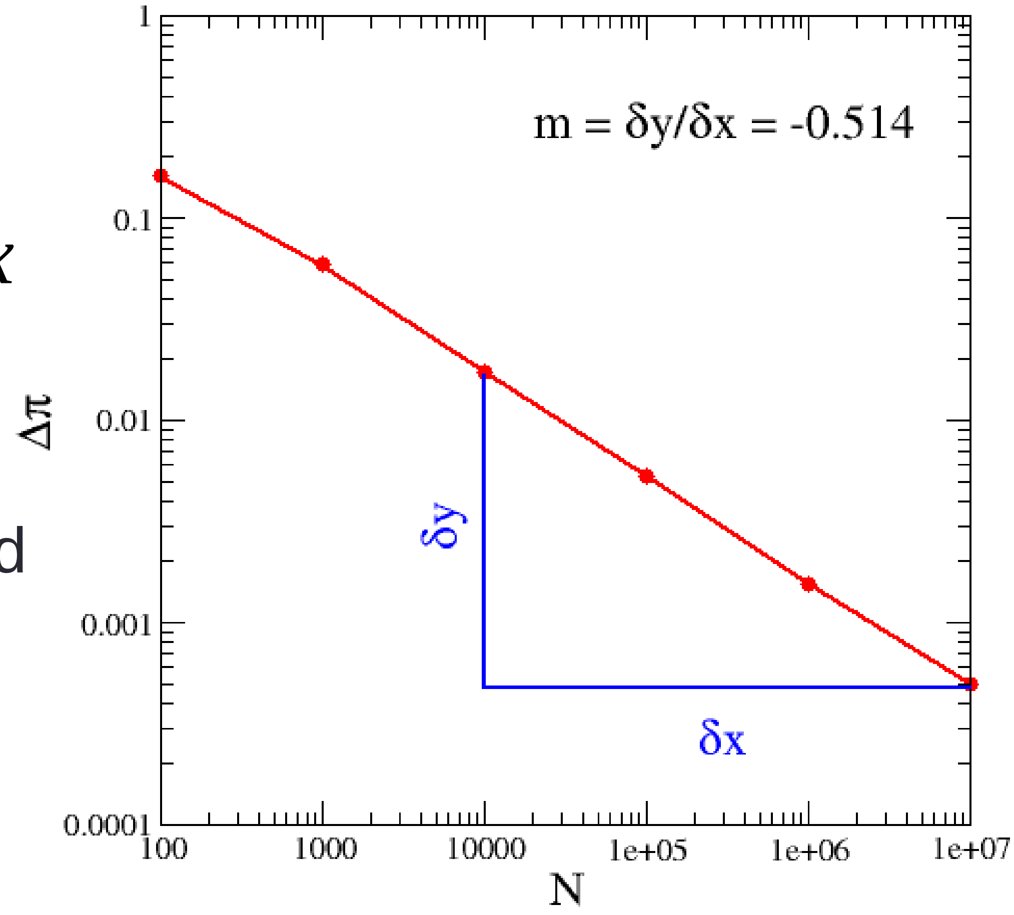
- Log-log plot

$$y = ax^b$$

$$\log y = \log a + b \log x$$

- Exponent b, is gradient
  - b ≈ -0.5
- Law of large numbers and central limit theorem

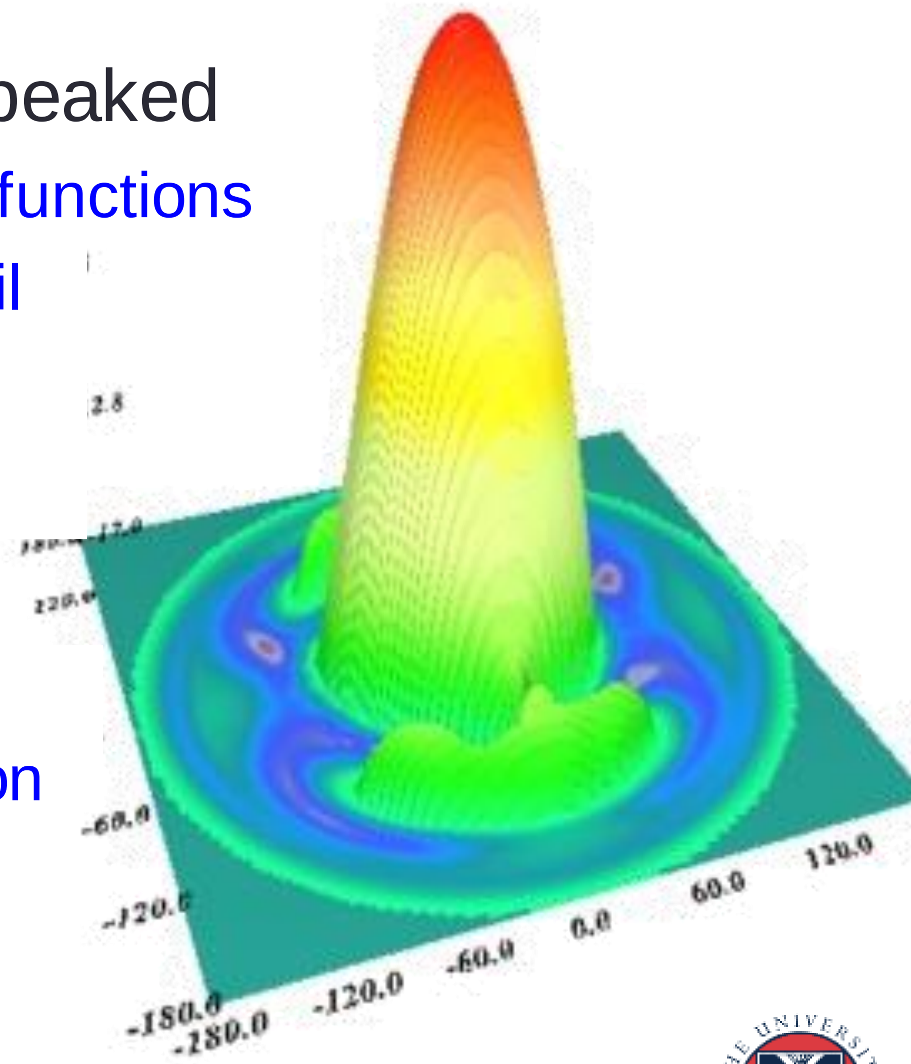
$$\boxed{\Delta \sim 1/\sqrt{N}}$$



True for *all* Monte Carlo methods

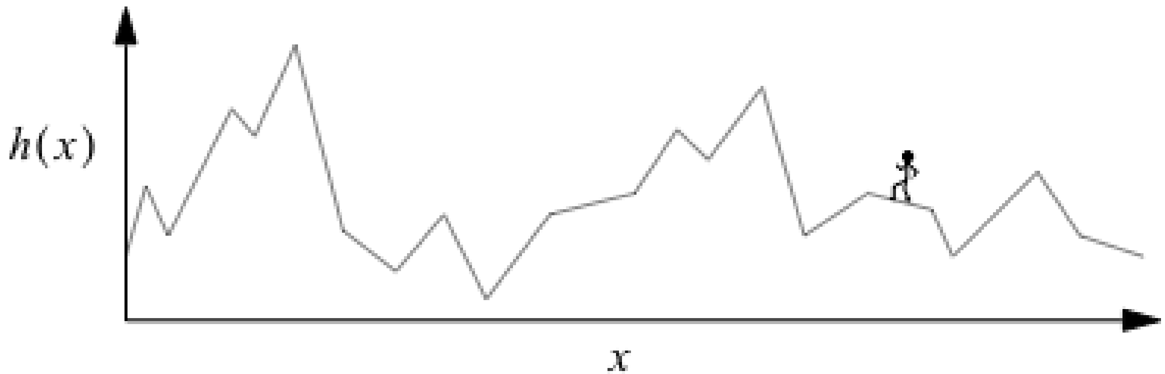
Importance Sampling

- Distribution often sharply peaked
  - especially high-dimensional functions
  - often with fine structure detail
- Random sampling
  - $p(x_i) \sim 0$ for most x_i
  - N large for fine structure
- Importance sampling
  - generate weighted distribution
  - proportional to probability



Hill-walking example

- Want to spend your time in areas proportional to height $h(x)$



  - walk randomly to explore all positions x_i
  - if you always head up-hill or down-hill
    - get stuck at nearest peak or valley
  - if you head up-hill or down-hill with equal probability
    - you don't prefer peaks over valleys
- Strategy
  - take both up-hill and down-hill steps but with a *preference* for up-hill

Markov Process

- Generate samples of $\{x_i\}$ with probability $p(x)$
  - x_i no longer chosen independently
- Generate new value from old: evolution

$$x_{i+1} = x_i + \delta x$$

- Accept/reject change based on $p(x_i)$ and $p(x_{i+1})$
  - if $p(x_{i+1}) > p(x_i)$ then accept the change (**uphill** move)
  - if $p(x_{i+1}) < p(x_i)$ then accept **with probability** $\frac{p(x_{i+1})}{p(x_i)}$ (**downhill** move)
- Asymptotic probability of x_i appearing is proportional to $p(x)$
- Need random numbers
  - to generate random moves δx and to do accept/reject step

AA Markov 1856-1922

Markov Chain Monte Carlo (MCMC)

- The generated samples form a ***Markov Chain***
  - e.g. the sequence of locations during your hill walk
  - new position generated from the old position
  - accept / reject step is called the ***Metropolis Algorithm***
- The update process must be ***ergodic***
  - able to reach all x
  - if the updates are non-ergodic then probability distribution will not be sampled correctly
- Takes some time to ***equilibrate***
  - starting point is random
  - need lots of updates to forget where you started from

Statistical Physics

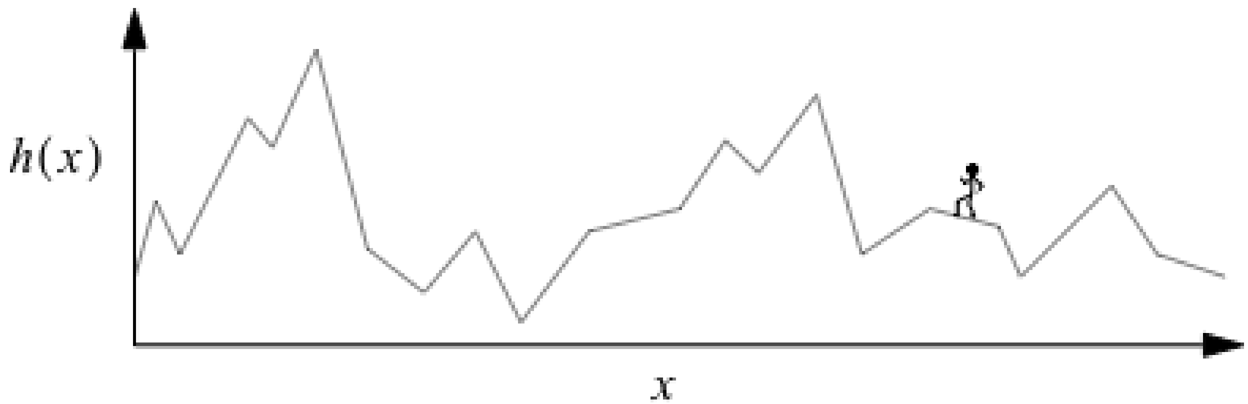
- Many applications use Markov Chain Monte Carlo
- Statistical physics is an example
- Systems have extremely high dimensionality
  - e.g. positions and orientations of millions of atoms
  - this is a multi-million-dimensional system
- Use MC to generate “snapshots” or configurations of the system
- Average over these to obtain answer
  - each individual state has no real meaning on its own
  - quantities determined as averages across all the states

Optimisation Problems

- Optima of function rather than averages
- Often minimise or maximise functions of many variables
  - minimum distance for travelling salesman problem
  - minimum cost function for machine learning / neural networks
- Procedure
  - take an initial guess
  - successively update to progress towards solution
- What changes should be proposed?
  - reduce/increase function with each update (steepest descent/ascent) …
  - … but this will only find the local minimum/maximum

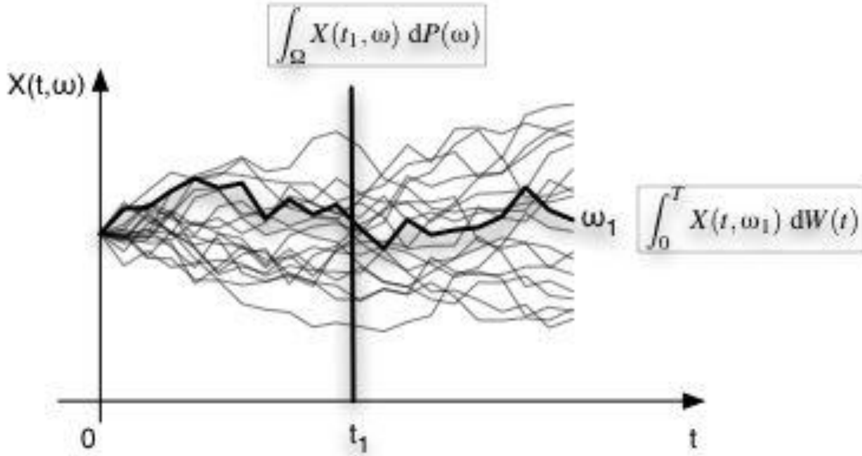
Stochastic Optimisation

- Add a random component to updates
  - e.g. ***Simulated Annealing***
- Sometimes make "bad" moves
  - possible to escape from local minima
  - but want more up-hill steps than down-hill ones
- Hill-walking example
  - find the highest peak in the Alps by maximising *h*(*x*)



MC in Finance

- Price model called Black-Scholes equation
  - Partial Differential Equation (PDE)
  - based on Geometric Brownian Motion (GBM) of underlying asset
    - this is a random process – use random numbers!
- Assumes a "perfect" market
  - markets are not perfect, especially during crashes!
  - many extensions
  - area of active research
- Use MC to generate many different GBM paths
  - statistically analyse ensemble



Numerical Weather Prediction

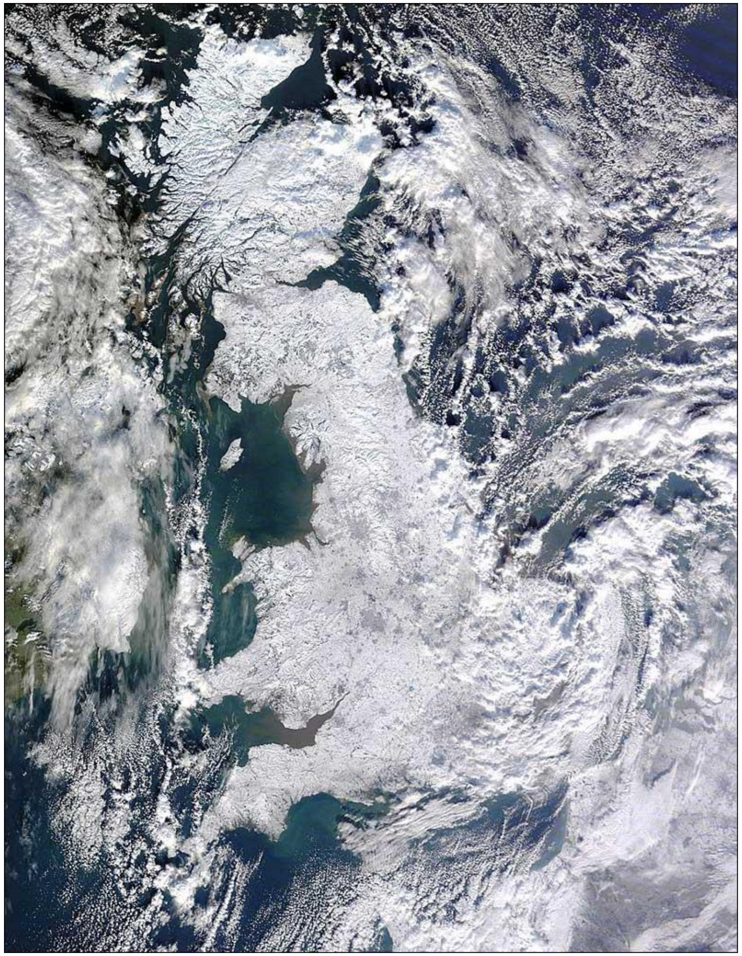


Image taken by NASA's Terra Satellite
7th January 2010

Britain in the grip of a very cold spell of weather

NWP in the UK

- Weather forecasts generated by the UK Met Office
  - code is called the ***Unified Model***
  - same code runs climate model and weather forecast
  - can cover the whole globe

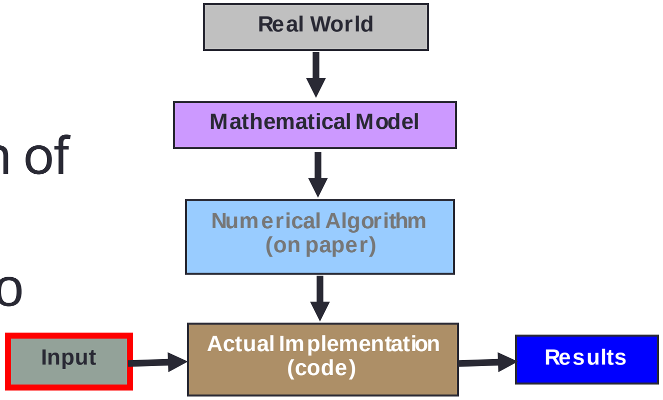
- Current supercomputer
  - Cray XC40
  - almost half a million processor-cores
  - weighs 140 tonnes

https://www.metoffice.gov.uk/about-us/what/technology/supercomputer

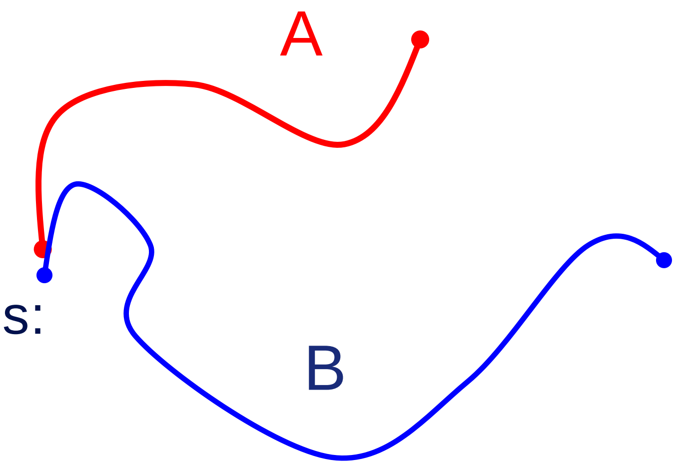
Initial conditions and the Butterfly effect

- The equations are extremely sensitive to initial conditions
  - small changes in the initial conditions result in large changes in outcome
- Discovered by Edward Lorenz circa 1960
  - 12-variable computer model
  - tiny variations in initial input parameters
  - totally different final weather patterns

- The Butterfly effect
  - "flap of a butterfly's wings can affect the path of a tornado"
  - my prediction is wrong because of effects too small to see



Chaos, randomness and probability



- A Chaotic system evolves to very different states from close initial states
  - no discernible pattern

- Use this to estimate how reliable our forecast is:
- Change the initial conditions a small amount
  - –based on uncertainty of measurement
  - –run a new forecast
- Repeat many times (random numbers to do perturbation)
  - –generate an “ensemble” of forecasts
  - –van then estimate the probability of the forecast being correct
- If we ran 100 simulations and 70 said it would rain
  - –probability of rain is 70%
  - –called **ensemble** weather forecasting

Parallelisation

- Real simulations use parallel computers

- Large simulations can require trillions of random numbers!
  - parallelisation introduces additional complexities …
- Run separate random number generators on each process
  - for speed of execution
- Ensure they are all given different seeds
  - so each process generates different random numbers
- Difficult to maintain reproducibility
  - e.g. what happens if you change the number of parallel processes?

Summary

- Random numbers used in many simulations
- Mainly to efficiently sample a large space of possibilities
- Different random numbers explore different possibilities
  - re-running with a different seed gives different answer
  - leads to a statistical uncertainty
- For MC simulation with N samples, error scales as $1/\sqrt{N}$
  - can control the error by choosing appropriate N
  - reducing error by factor of 10 takes 100 times longer!