Monte Carlo methods

What is MC?

- anything which involves a random number generator
- in essence, a probability based brute force numerical approach
- (inelegant)
- computationally inefficient
- EASY (for coding and conceptually)
- robust
- tractable

- crude code can be developed without need for reference to complex algorithms or numerical textbooks

- can become very sophisticated for most challenging problem

Computer time is cheaper than your time!

Applications in astronomy (observational)

- MC realizations of synthetic data (images, spectra etc.) Test reduction software, quantify errors, bias etc.

- Design of experiments/observational campaigns: justify S/N, sample size etc. for science goal. Demonstrate feasibility.

- Analysis of data – surveys, detection thresholds, test population models, parameter estimation (MCMC).

Applications in astronomy (theoretical)

- Rapid numerical technique (few lines of code). Perhaps inefficient - but easy to use.

- e.g. Solution of 2/3-D transfer problems, dust scattering, polarization

- e.g. Hierarchical structure formation (merger histories for SMBHs)

- e.g. Population synthesis

MC simulations – two types in literature

1) Realistic simulations
- Entirely valid numerical procedure if, as N->infty, the basic equations and boundary conditions are restored.

- Will be true if MC rules were obtained from physics of phenomenon (true for many transfer problems)

2) Toy simulations
- Plausible rules adopted to mimic aspects of phenomenon (real physics poorly understood or too complex – eg turbulence, star formation)

- More reliably, “rules” may be adopted based on large realistic simulation and used to explore parameter space

E.G. Star formation, SN in galaxy evolution codes.

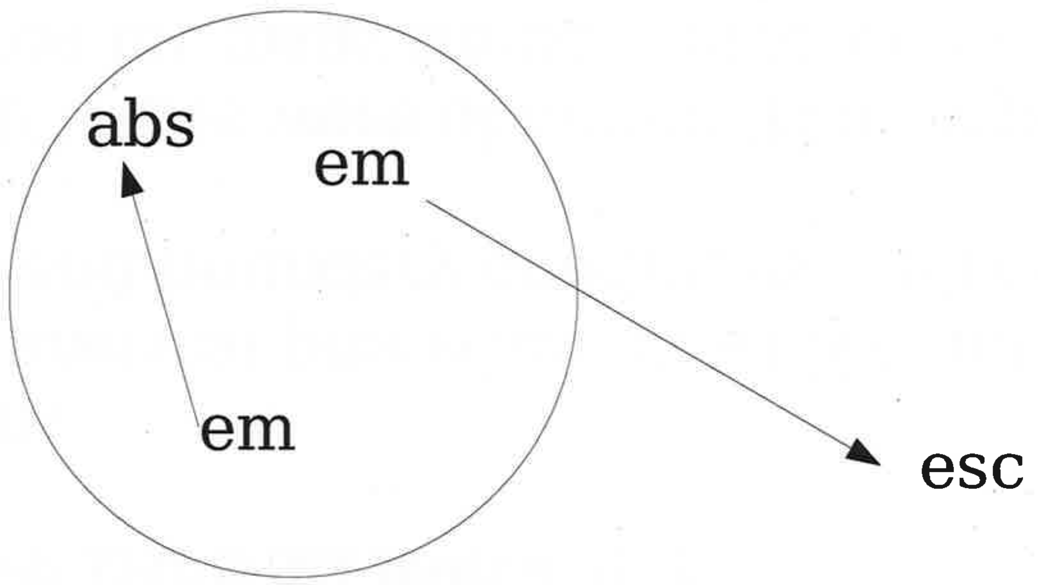
Test problem: escape probability from a uniform sphere

To illustrate MC methods let's work through a simple problem using a numerical MC approach.

Consider a uniform sphere of radius R with constant emissivity and absorption (κ) coefficients and zero scattering coefficient. Calculate $p_{esc}(\tau_{tot})$, the average escape probability for photons emitted within the sphere, as a function of the optical depth:

$$\tau_{tot} = \kappa R$$

This problem has an analytic solution (see Osterbrock)



MC approach – the plan!

0) choose MC quanta (energy bundles – not obliged to use Nature's quantization)

1) randomly place energy bundles in sphere (from initial pdf)

2) randomly assign directions to bundles (from initial pdf)

3) compute distance (d) to edge of sphere

4) randomly decide whether bundle is absorbed before it travels the distance d (use interaction pdf)

5) record fraction that make it out

pdf = probability distribution function

Random numbers

C, FOTRAN etc. have in built random number generators. A computer cannot generate truly random numbers (in the way that a physical process can) and so our definition of a random number generator has to be a bit vague. From Numerical Recipes:

“A working, though imprecise, definition of randomness in the context of computer-generated sequences, is to say that the deterministic program that produces a random sequence should be different from, and – in all measurable respects – statistically uncorrelated with, the computer program that uses its output. In other words, any two different random number generators ought to produce statistically the same results when coupled to your particular application program.”

Linear congruential generators

Most common way to generate uniform deviates (i.e. Random numbers in specified interval, often 0->1). System-supplied random number generators are usually of this sort and typically involve a seed (integer) and then calls to function:

```
srand(1084515760)

x = rand()/(RAND_MAX + 1)
```

Ideally, the seed gives a unique, infinite string of numbers through which we travel on each call to rand(). Realistically, it should put us at an unpredictable point in a very long sequence of uncorrelated numbers.

With the same seed, you will get exactly the same sequence of numbers following – beware! (For some applications it is wise to take the seed from the system clock.)

Linear congruential generators

The recursive algorithm to obtain the pseudo random number sequence for system RNGs is usually something like

$$I_{j+1} = a\, I_j + c \pmod{m}$$

Although very sensitive to the choice of the constants, this can generate strings of numbers with very long periods. The "minimal standard" generator (the least good one you should ever consider using) has

$$a = 16807$$
$$c = 0$$
$$m = 2147483647$$

It requires numbers greater than 32-bit machines can cope with – even this simplest method requires assembly language tricks!

Linear congruential generators

The “minimal standard” is known to be imperfect:

1) a scatter plot in multi-dimensional space leads to banding

2) high-order bits (most sig.) are more random than low-order

Various improved (but slower) algorithms exist. Most strongly recommended is probably the numerical recipes “ran2” routine.

Morals

1) Don't interfere with random number generator algorithm unless you really know what you are doing.

2) Always be sure you are using a sufficiently high quality algorithm.

3) Never re-use a rand() call and be careful with re-using seeds.

Linear congruential generators

A simple test for RAN2:

Calculate 20,000 random numbers and bin them into 20 equal intervals (0.00,0.05), (0.05,0.10)...(0.95,1.00).

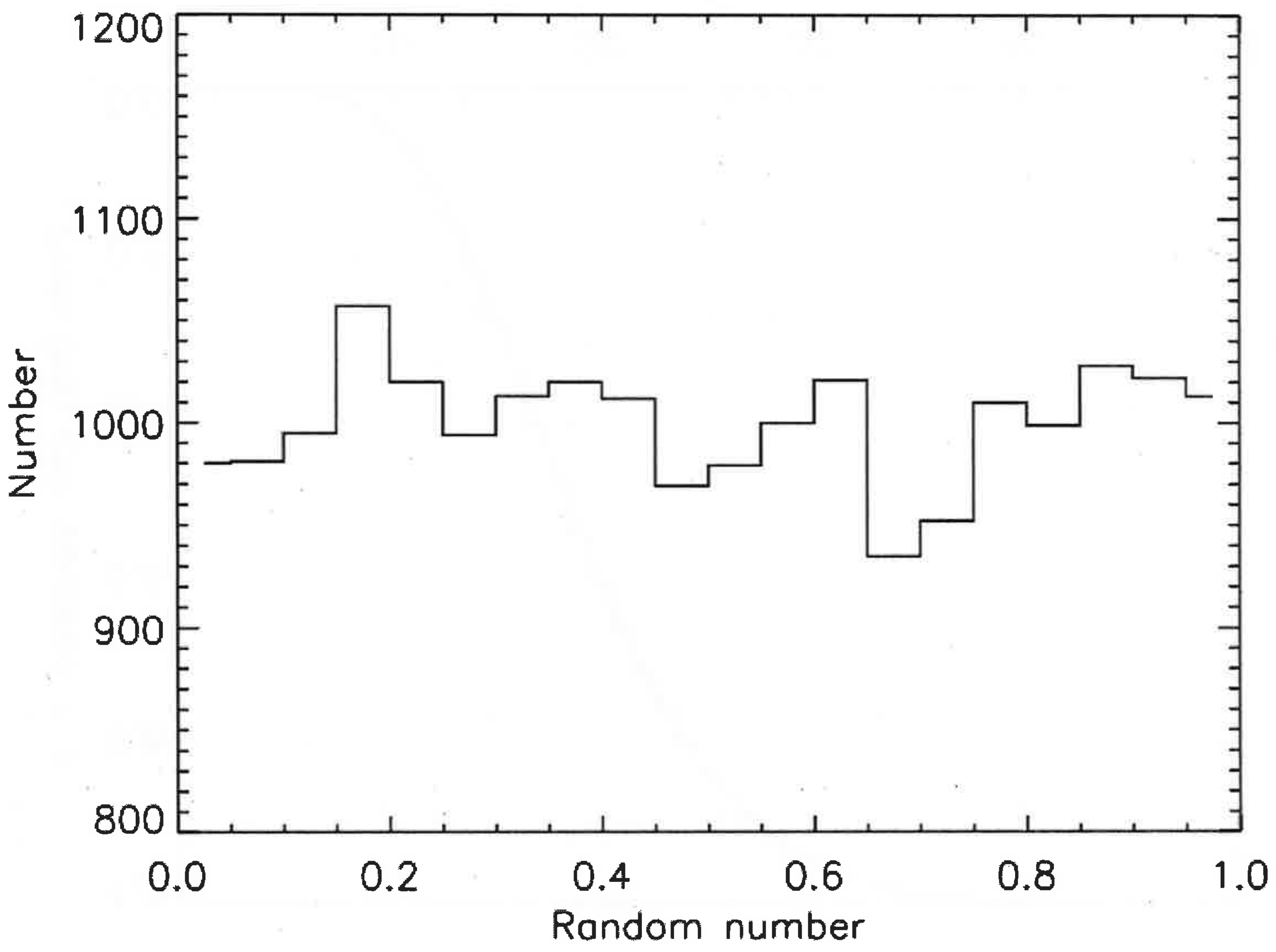
Count how many lie in each bin (n_i) and compare to the expected value of 1000.

Compute the statistic

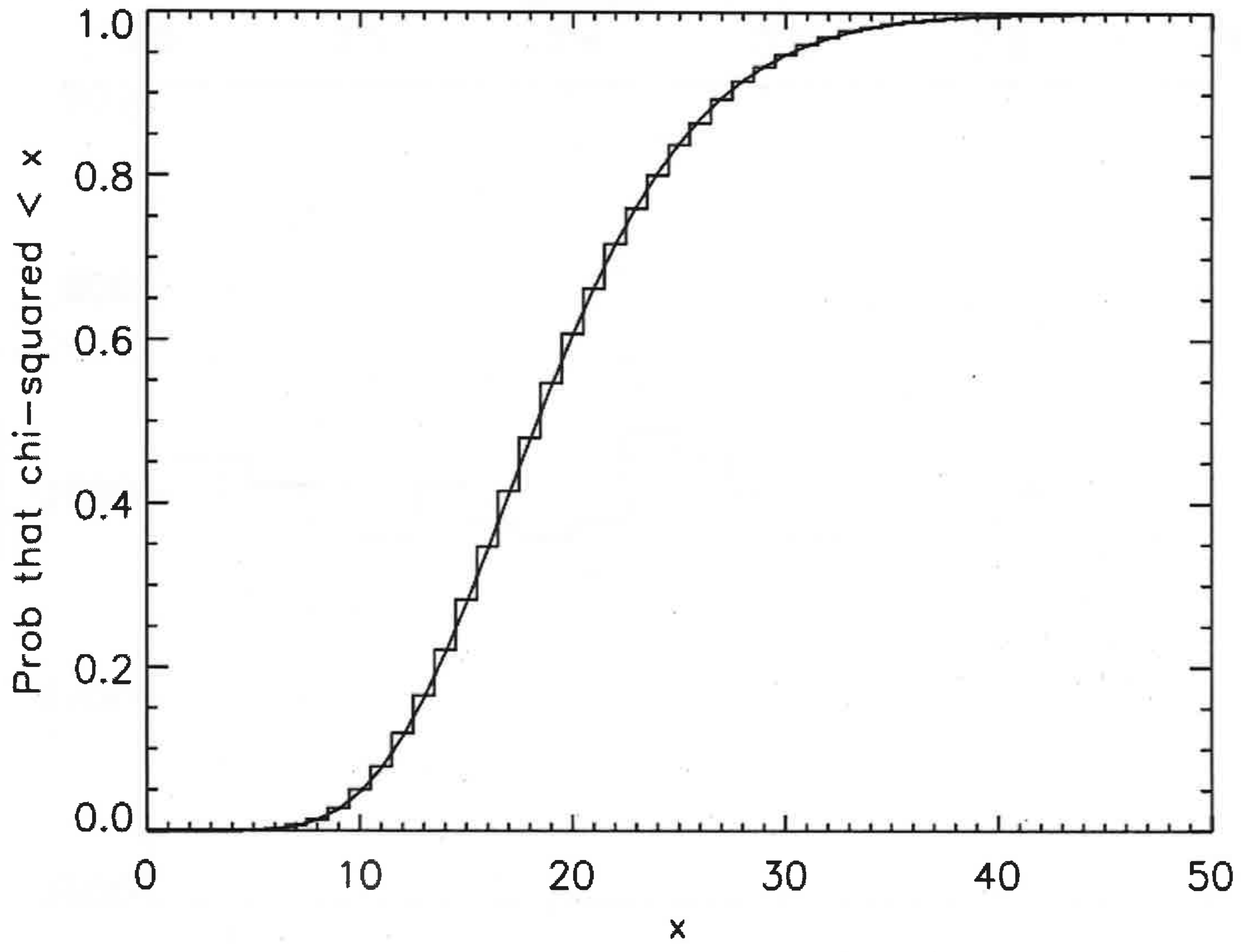
$$\chi^2 = \Sigma_i (n_i - 1000)^2 / 1000$$

Repeat this process 10,000 times and compare the distribution of χ^2 values with that expected.

Results for 1 set of 20,000 calls



Results from 10,000 sets of 20,000 calls



MC approach – the plan (revisited 1)!

0) choose MC quanta (energy bundles – not obliged to use Nature's quantization)

1) randomly place energy bundles in sphere (from initial pdf)

2) randomly assign directions to bundles (from initial pdf)

3) compute distance (d) to edge of sphere

4) randomly decide whether bundle is absorbed before it travels the distance d (use interaction pdf)

5) record fraction that make it out

pdf = probability distribution function

Probability distribution functions

Our RNG gives us a uniform distribution in the interval 0->1 but in general this is not what we want. Eg in our test case we want to place the energy bundles at random in a uniformly emitting spherical volume (0<r<R).

To convert from our uniformly distributed random numbers (call them z) to a non-uniformly distributed variable (such as r) is very simple. Let the (known) probability distribution function for the variable we want (r) be p(r). Then we require

$$p(r)\,dr = p(z)\,dz$$

where p(z) = 1 is the uniform probability distribution for the random number z. Integrating gives us the relationship between the random number and r

$$\int_0^r p(r)\,dr = z$$

Probability distribution functions (for step 1)

"1) randomly place energy bundles in sphere (from initial pdf)"

For our example, the probability distribution function (for radial position in the uniform sphere) must be

$$p(r) = (4 \pi r^2) / (4/3 \pi R^3)$$

and therefore the relationship between our random numbers and the positions of the bundles is

$$r = R z^{1/3}$$

What about the direction of the photon bundle? Since it is only a 1-D problem we need to know only the angle between the radial direction and the direction of propagation. What is the probability distribution for this angle, assuming isotropic emission?

Probability distribution functions for step (2)

“2) randomly assign directions to bundles (from initial pdf)”

Recall that isotropic emission means that the direction vectors are spread uniformly over the unit sphere. Therefore the probability distribution in solid angle must be uniform

$$p(\Omega)\, d\Omega = d\Omega \,/\, (4\pi)$$

Using

$p(\Omega)\, d\Omega = p(\theta)\, p(\varphi)\, d\theta\, d\varphi$ and $d\Omega = \sin\theta\, d\theta\, d\varphi$

it is clear that we require

$$p(\varphi) = 1 \,/\, (2\pi) \quad \text{and} \quad p(\theta) = \sin\theta \,/\, 2$$

Leading to (for uniformly distributed random number, z)

$$\theta = \cos^{-1}(1 - 2z)$$

MC approach – the plan (revisited 2)!

0) choose MC quanta (energy bundles – not obliged to use Nature's quantization)

1) randomly place energy bundles in sphere (from initial pdf)

2) randomly assign directions to bundles (from initial pdf)

3) compute distance (d) to edge of sphere – cosine rule

4) randomly decide whether bundle is absorbed before it travels the distance d (use interaction pdf)

5) record fraction that make it out

pdf = probability distribution function

Probability distribution (for step 4)

“4) randomly decide whether bundle is absorbed before it travels the distance d (use interaction pdf)”

Let's say that the distance to the edge of the sphere is “d”. Then the optical depth to the edge is

$$\tau_{edge} = \kappa\, d$$

We can randomly choose the optical depth (τ) an energy bundle travels before interaction using the well known fact that

$$I = I_0\, e^{-\tau}$$

which translates to a probability distribution of

$$p(\tau) = e^{-\tau}$$

Probability distribution (for step 4)

As before, this pdf ($p(\tau) = e^{-\tau}$) can be used to generate values of τ from a random number z. Applying the integral as above one obtains

$$\tau = - \ln (1 - z)$$

which is equivalent to

$$\tau = - \ln z$$

Having chosen τ, the optical path our packet can pass without absorption we simply compare with τ_{edge} :

if $\tau > \tau_{edge}$ then it escapes

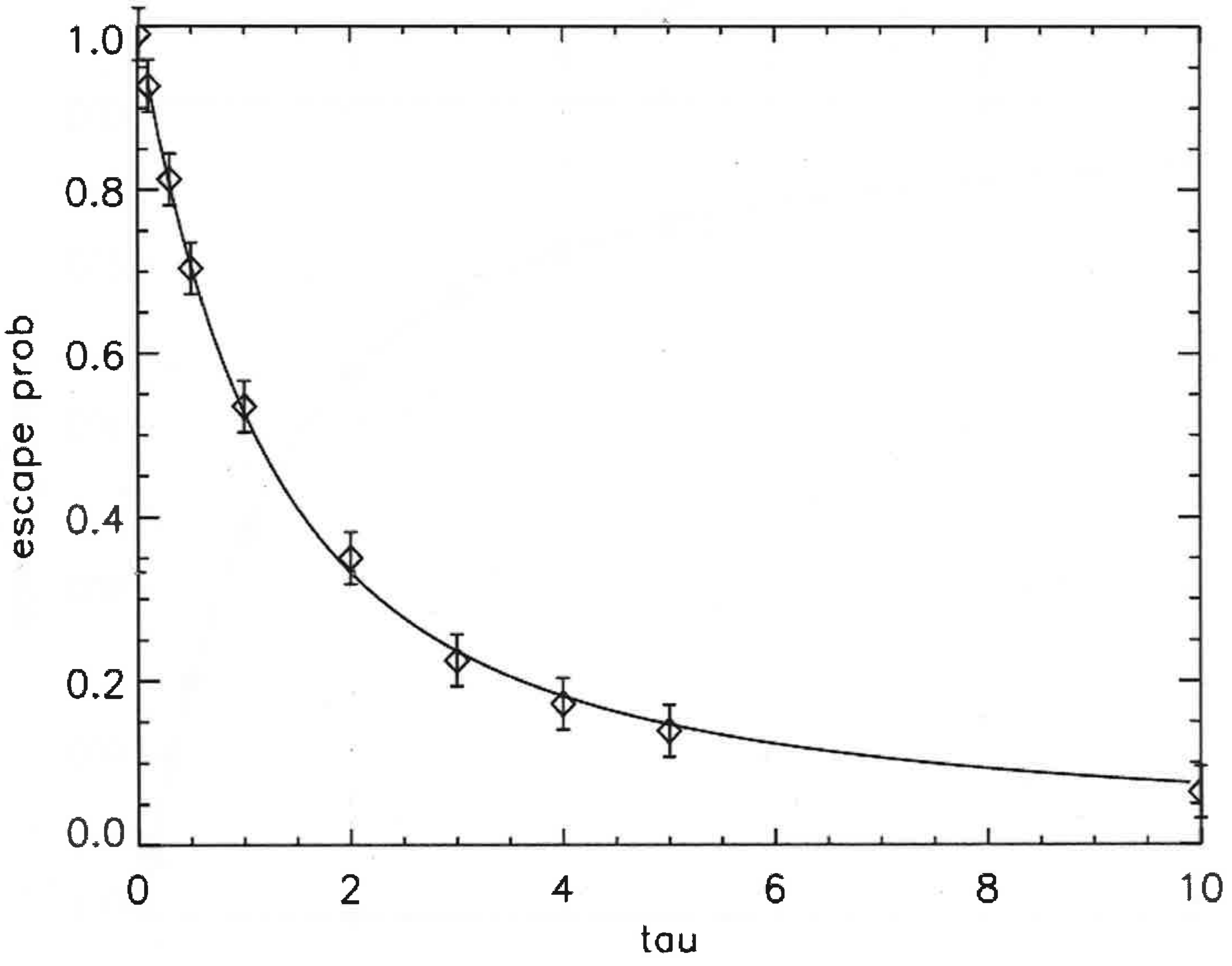
if $\tau < \tau_{edge}$ then it is absorbed and does not escape

Repeat (step 5)

We can now generate packets and decide whether they escape. All we need to do now is repeat this many many times and count the fraction of the packets that escape. That fraction is then the escape probability.

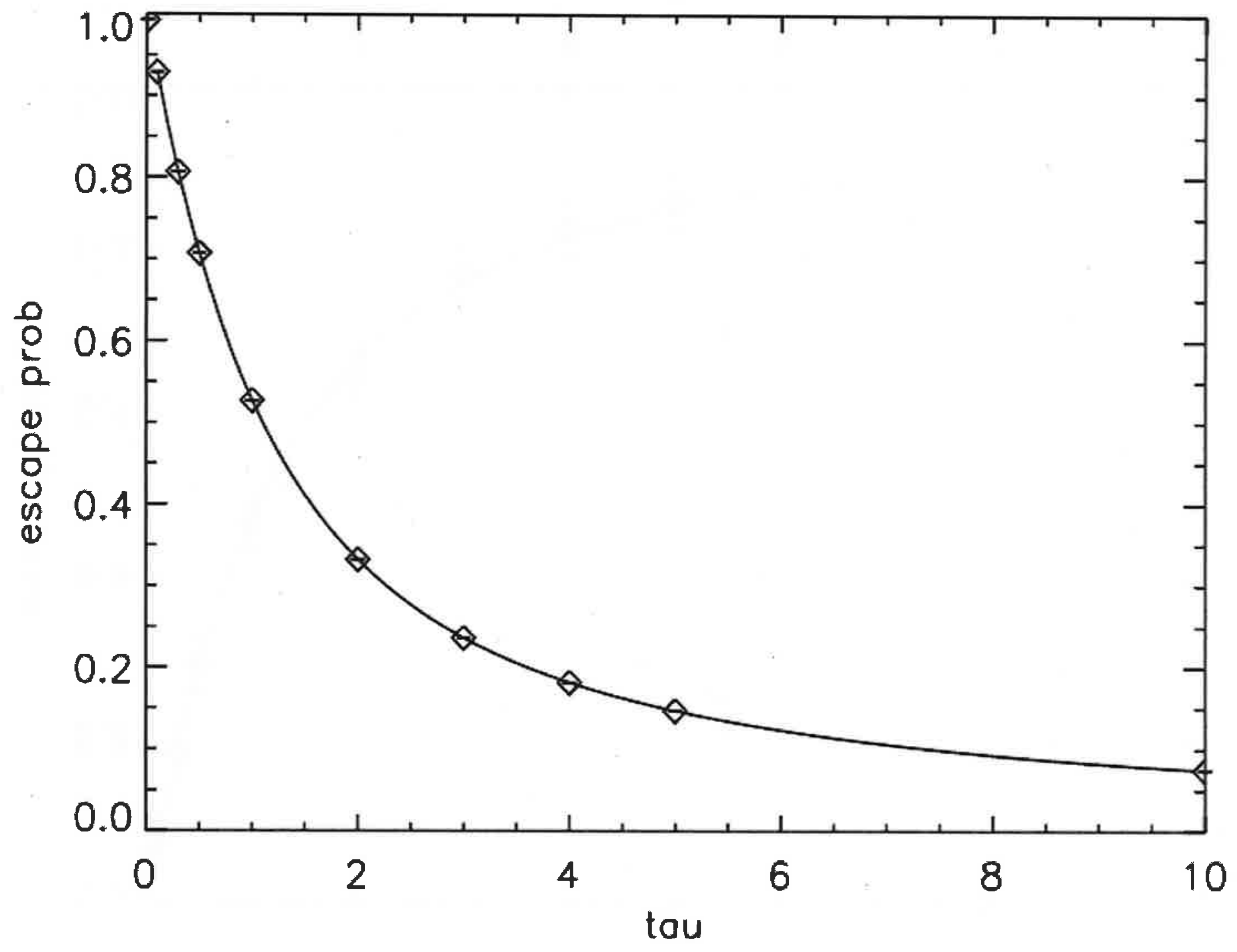
As a test, can do this for several different values of τ and compare with analytic result.

Results (1000 packets)



$$p_{esc} = 0.75\tau^{-1}\left[\,1 - 0.5\tau^{-2} + (\tau^{-1} + 0.5\tau^{-2})e^{-2\tau}\,\right]$$

Results (100000000 packets)



More sophisticated problems

For the spherical escape problem an analytical results was available – of course that is preferable to a numerical result.

But the problem does not need to become significantly more complex before analytic results are very difficult (or impossible) to obtain.

E.g. Escape probability from uniformly emitting non-spherical geometry (e.g. Cone or cylinder)

...or allow for the (physically important) process of scattering

Exercise for next time...

Write your own MC code (C, FORTRAN – whatever you like) to compute the escape probability from a uniformly emitting sphere in which the gas can both scatter and absorb electromagnetic energy. The total optical depth from the centre of the sphere to the edge will be

$$\tau = (\sigma + \kappa)$$

Assume that scattering and absorption are equally likely (i.e. $\sigma = \kappa$). Assume that scattering is isotropic.

Start by writing your own code for the absorption only case and check that it works. Then try to add the scattering. Bring along whatever results you have (and a printout of your code) for next time.

Alternative exercise for next time...

Look on astro-ph or ADS and find 2+ papers that use MC methods (for something other than transport theory). Read the papers and come along next time ready to give a 5 min description of them with emphasis on why they used MC and what they did (or did not!) achieve by doing so.

...also, if you have an idle moment

Write a (very very very simple) MC code to compute the value of π (using only the random number generator and arithmetical operations – no trig functions etc!). See how many sig. fig. you can get correct in 10min of computer time.

Summary

- “Monte Carlo methods” encompass a broad range of numerical techniques which exploit random numbers.

- Usually they are relatively simple to use and very easy to code (perhaps inefficient but CPU time is cheap).

- Random number generators must be treated with care (and respect). Be sure your generator is sufficiently good for your needs.

- Basic building blocks of code will involve conversion of uniform probability distribution (random number) to distribution for some other quantity.

- Simple examples provide handy tests for understanding and coding.