Numerical Methods for Astrophysics:

RANDOM NUMBERS

Michela Mapelli

Random numbers. Concept

Random numbers are ubiquitous in physics/astrophysics:

– some (astro)physical process is intrinsically random
 (e.g. exact moment of a radioactive decay is random)

– other (astro)physical quantities are not intrinsically random
 but we might need random numbers to represent them

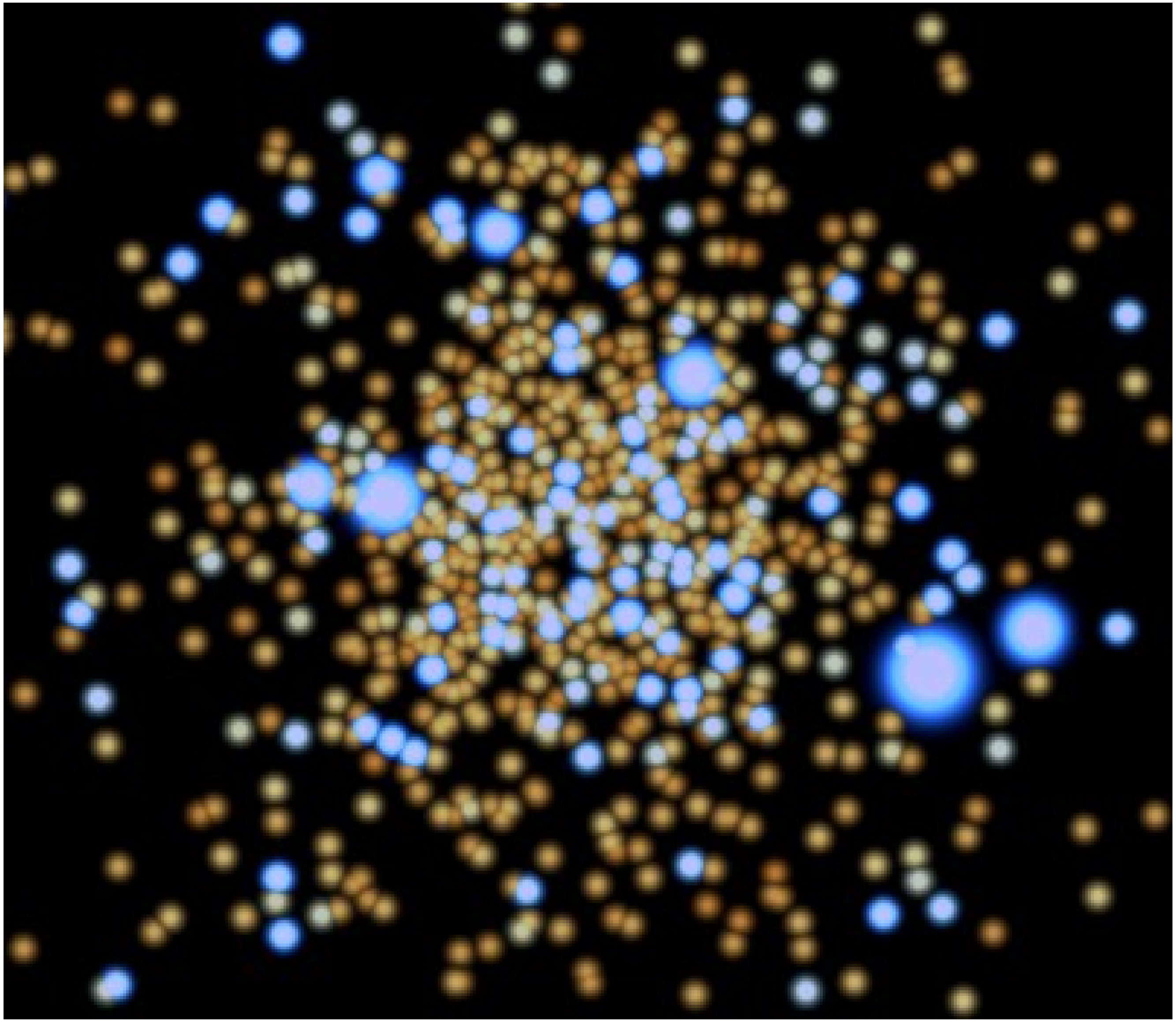
EXAMPLEs:

*** produce mock samples of astrophysical data,**
 e.g. magnitudes of stars in a star cluster

*** in computational astrophysics, initial conditions of simulations are often generated through random numbers**
 e.g. N-body model of a galaxy or star cluster
  Initial star positions can be generated
   - on a fixed grid (unnatural..)
   - randomly drawing initial positions
    from distribution functions (more natural)

Random numbers. Concept

Example: initial conditions for a simulation of a star cluster



Random numbers. Random generators

Is a computer able to generate genuine random numbers?
NO, just PSEUDO-RANDOM numbers, generated with a formula

$$x' = (a\,x + c) \bmod m$$

LINEAR CONGRUENTIAL RANDOM NUMBER GENERATOR

a, c, m **= integer constants**
x **= integer variable**

take *x*’ and plug it back onto the right-hand side of the equation
→ generates a series of numbers

In python:

```
#file examples/random/rand_gen.py
N = 100
a = int(1664525)
c=int(1013904223)
m=int(4294967296)
x=1
results = []
for i in range(N):
      x = (a*x+c)%m
      results.append(x)
print(results)
```

Random numbers. Random generators

$$x' = (a\,x + c) \bmod m$$

LINEAR CONGRUENTIAL RANDOM NUMBER GENERATOR

a, c, m **= integer constants**
x **= integer variable**

If we use the same *a, c, m* and the same first *x*,
we will generate always the same series
→ guarantees REPRODUCIBILITY of scientific experiments

Note that if we generate N random numbers with N>*m*
the *m*+1 number will be the same as the 1st number
the *m*+2 number will be the same as the 2nd number
etc etc

i.e. THE SEQUENCE REPEATS FROM THE BEGINNING
→ WARNING: terrible mistake, make sure that N<*m*

Random numbers. Random generators

Random generators in python:

*** random package**

random.random() generates floating point random numbers between 0 and 1

To obtain a random between min and max do

```
a=random.random()
b=a * (max – min) + min
```

random.randint(min, max) generates integer random numbers between min and max

*** numpy.random package**

numpy.random.rand() generates floating point random numbers between 0 and 1

To obtain a random between min and max do

```
a=numpy.random.rand()
b=a * (max – min) + min
```

See examples/random/use_random.py
See examples/random/use_nprandom.py

Random numbers. Random seed

First number of the series (first *x* in the linear congruential generator)

Uniquely determines the entire series

Default is computer clock,
but better set by hand to ensure reproducibility

with random.seed:

```
#examples/random/use_seed.py
from random import random, seed

seed(42) #assign 42 as seed
for i in range(10):
    a=random()
    print(a)
```

Random numbers. Random seed

First number of the series (first *x* in the linear congruential generator)

Uniquely determines the entire series

Default is computer clock,
but better set by hand to ensure reproducibility

with numpy.random.seed:

```
#examples/random/use_npseed.py
from numpy.random import random, seed

seed(42) #assign 42 as seed
for i in range(10): #calculate 10 random numbers with a loop
    a=random()
    print(a)

seed(42) #assign 42 as seed
b=random(10) #calculate 10 random numbers
             #with properties of numpy arrays
print(b)
```

Random numbers. Uniform deviates

Random numbers generated by a random generator are uniform deviates: each of them has the same probability to be generated within a given range.

In math. words,
the probability distribution function is constant over the range

$$p(x)\,\mathrm{d}x = \mathrm{const}\,\mathrm{d}x$$

probability distribution function

probability to draw a random number between x and x+ dx

normalization constant

$$\int p(x)\,\mathrm{d}x = 1$$

Random numbers. Non-uniform deviates

For a general (astro)physical problem, more likely that we need random numbers generated according to a non-constant probability distribution function

e.g. errors of measure follow Gaussian distribution

To generate non-uniform deviates:

- **first generate a set of uniform deviates**
- **then transform these uniform deviates into non-uniform deviates thanks to the laws of probability**

At least two techniques:

- **INVERSE RANDOM SAMPLING**
- **REJECTION METHOD**

Random numbers. Inverse random sampling

Fundamental transformation law of probabilities:

$$p(y)\,\mathrm{d}y = q(x)\,\mathrm{d}x$$

Probability p(y) dy of generating a number between y and y+dy
equal to probability q(x) dx of generating a number between x and x+dx
provided that both p(y) and q(x) are properly defined, i.e.

$$\int_{y_{min}}^{y_{max}} p(y)\,\mathrm{d}y = 1 \qquad \int_{x_{min}}^{x_{max}} q(x)\,\mathrm{d}x = 1$$

Hence y and x are related by a function y = y(x)

If x is a uniform deviate between 0 and 1, then q(x) = 1 and $\int_0^x \mathrm{d}x' = x$

Thus, for a generic p(y)

$$\boxed{\int_{y_{\min}}^{y(x)} p(y')\,\mathrm{d}y' = \int_0^x \mathrm{d}x' = x}$$

→ it is possible to generate a non-uniform random deviate
from a uniform random deviate

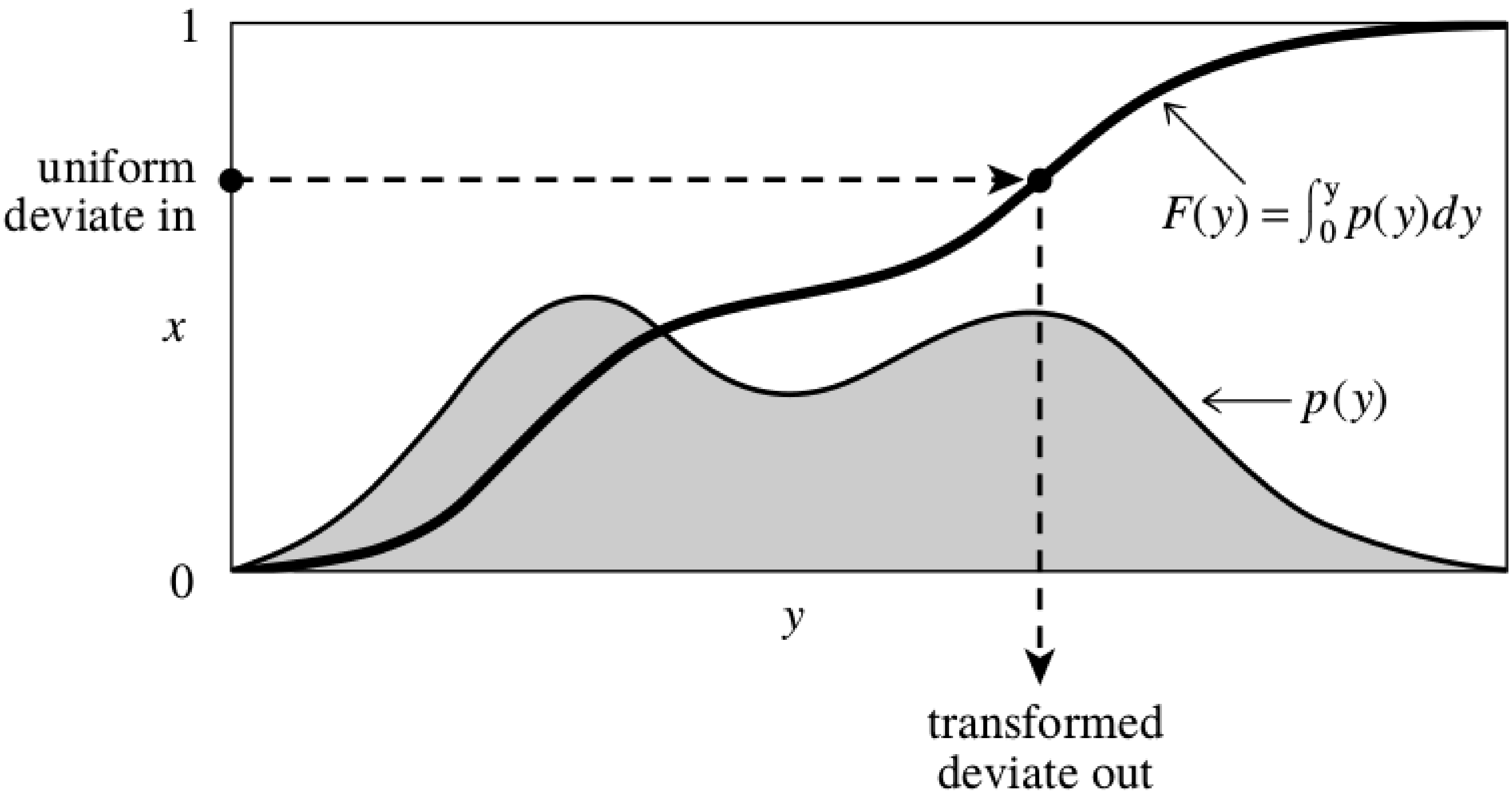
Necessary condition to extract y from x: that we can solve
the left-hand term of the integral and that we can solve y(x)

Random numbers. Inverse random sampling

Example:

$$\int_0^{y(x)} 2\, y' \, \mathrm{d}y' = x$$

$$\longrightarrow \quad y(x) = x^{0.5}$$



Random numbers. Inverse random sampling

Steps to produce non-uniform deviates with inverse random sampling:

1. **Take a probability distribution function p(y) of the quantity y you want to sample.**

2. **Integrate p(y) dy over the range to obtain the cumulative probability distribution function**

$$P(y) = \int_{y_{\min}}^{y} p(y')\mathrm{d}y'$$

3. **P(y) is monotonic and takes values from 0 to 1 by definition of probability.**

4. **Randomly sample the values x = P(y) of the cumulative distribution function between 0 and 1 (with a random generator).**

5. **Invert the function P(y) to get y = $P(y)^{-1}$**

6. **Repeat steps 4 and 5 as many times as you need to get y for N random numbers.**

Random numbers. EXERCISE, Salpeter:

EXERCISE: use the inverse random sampling to generate the masses of stars in a star cluster

The **Salpeter mass function (Salpeter1955)** is one of the most popular initial mass functions for stars. It is defined as

$$p(m)\,\mathrm{d}m = \mathrm{const}\, m^{-\alpha}\,\mathrm{d}m$$

where $\alpha = 2.3$.

Given a population of young stars (possibly in the zero-age main sequence), the probability to have a star of mass *m* in this population is $p(m) = \mathrm{const}\ m^{-\alpha}$
Massive stars are significantly less common than light stars.

Assuming that the minimum stellar mass is m_{min}= 0.1 Msun and
the maximum stellar mass is m_{max} = 150 Msun,
randomly calculate the mass of 10^6 stars distributed according to the Salpeter initial mass function by using the inverse random sampling technique.
Plot the resulting population of stellar masses with an histogram.

Suggestion: First you have to calculate the normalization constant const.

Salpeter: $\phi(m)\,dm = \text{const}\; m^{-\alpha}\,dm$

① Find constant $\int_{m_{min}}^{m_{max}} \phi(m)\,dm = 1$

$$1 = \text{const} \int_{m_{min}}^{m_{max}} m^{-\alpha}\,dm = \text{const}\,\frac{1}{(1-\alpha)}\, m^{1-\alpha}\Big|_{m_{min}}^{m_{max}} =$$

$$= \frac{\text{const}}{(1-\alpha)}\left(m_{max}^{1-\alpha} - m_{min}^{1-\alpha}\right)$$

$$\Longrightarrow \text{const} = \frac{1-\alpha}{\left(m_{max}^{1-\alpha} - m_{min}^{1-\alpha}\right)}$$

② Calculate cumulative PDF :

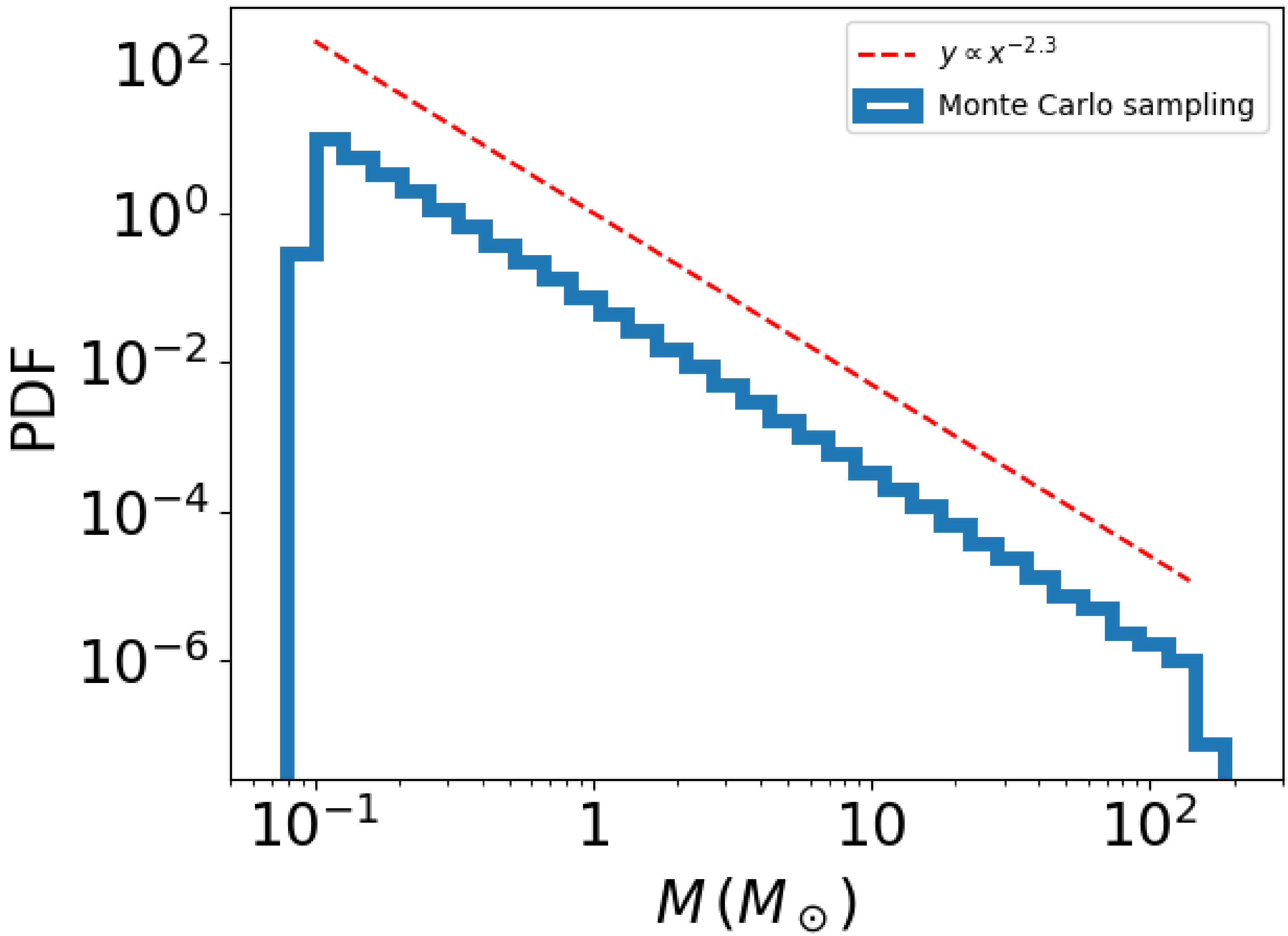
$$X = \text{const} \int_{m_{min}}^{m} \tilde{m}^{-\alpha} \, d\tilde{m} = \text{const} \, \frac{\tilde{m}^{1-\alpha}}{1-\alpha} \Big|_{m_{min}}^{m} =$$

$$= \frac{\text{const}}{1-\alpha} \left(m^{1-\alpha} - m_{min}^{1-\alpha} \right) = \frac{\left(m^{1-\alpha} - m_{min}^{1-\alpha} \right)}{\left(m_{max}^{1-\alpha} - m_{min}^{1-\alpha} \right)}$$

$$\Longrightarrow m^{\cancel{1-\alpha}} = \left[\left(m_{max}^{1-\alpha} - m_{min}^{1-\alpha} \right) X + m_{min}^{1-\alpha} \right]^{\frac{1}{1-\alpha}}$$

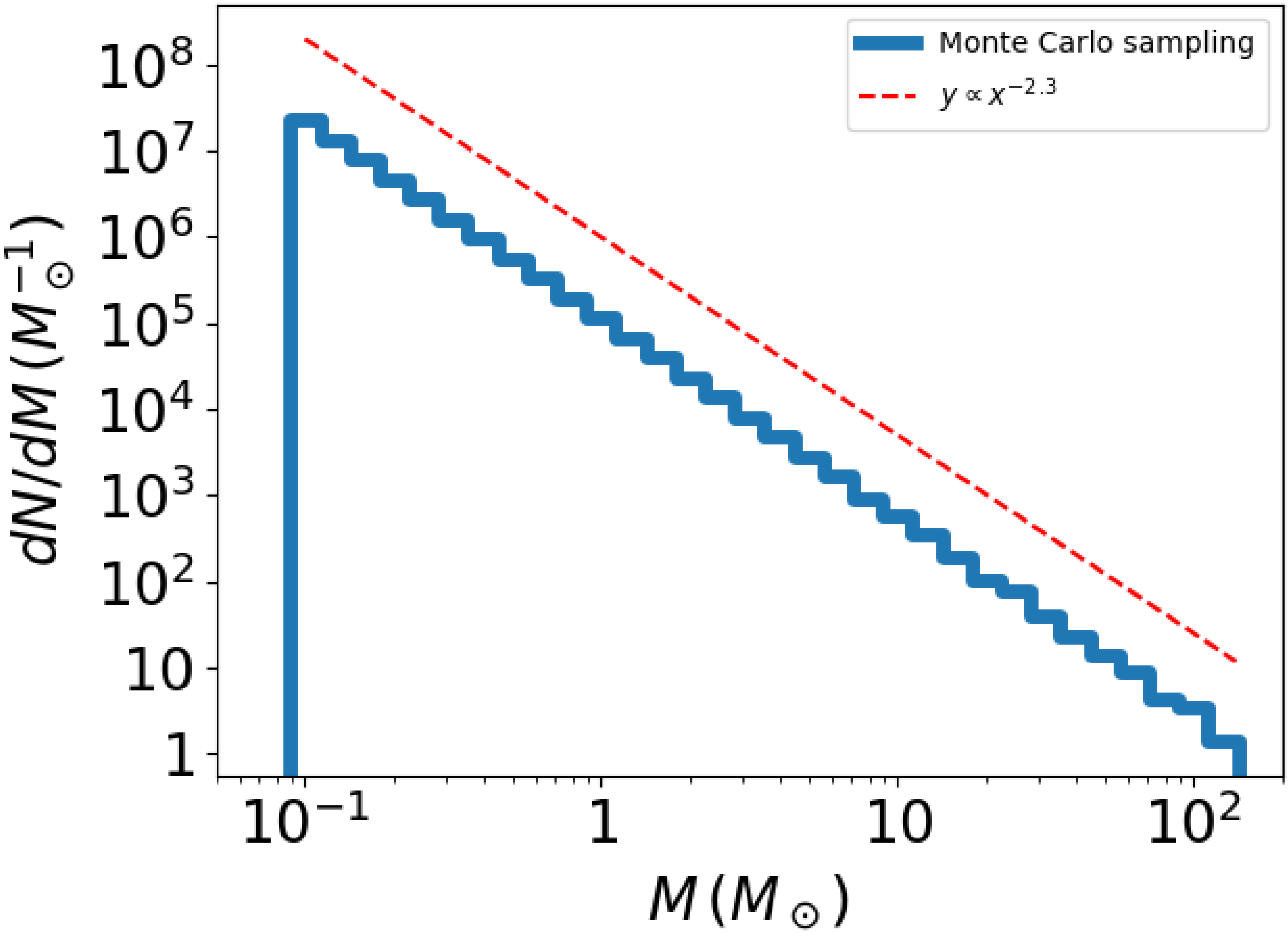
↑ uniform random number (X)

Random numbers. Inverse random sampling



The y axis looks like this if you use the option “density = True” of plt.hist(), which calculates the PDF

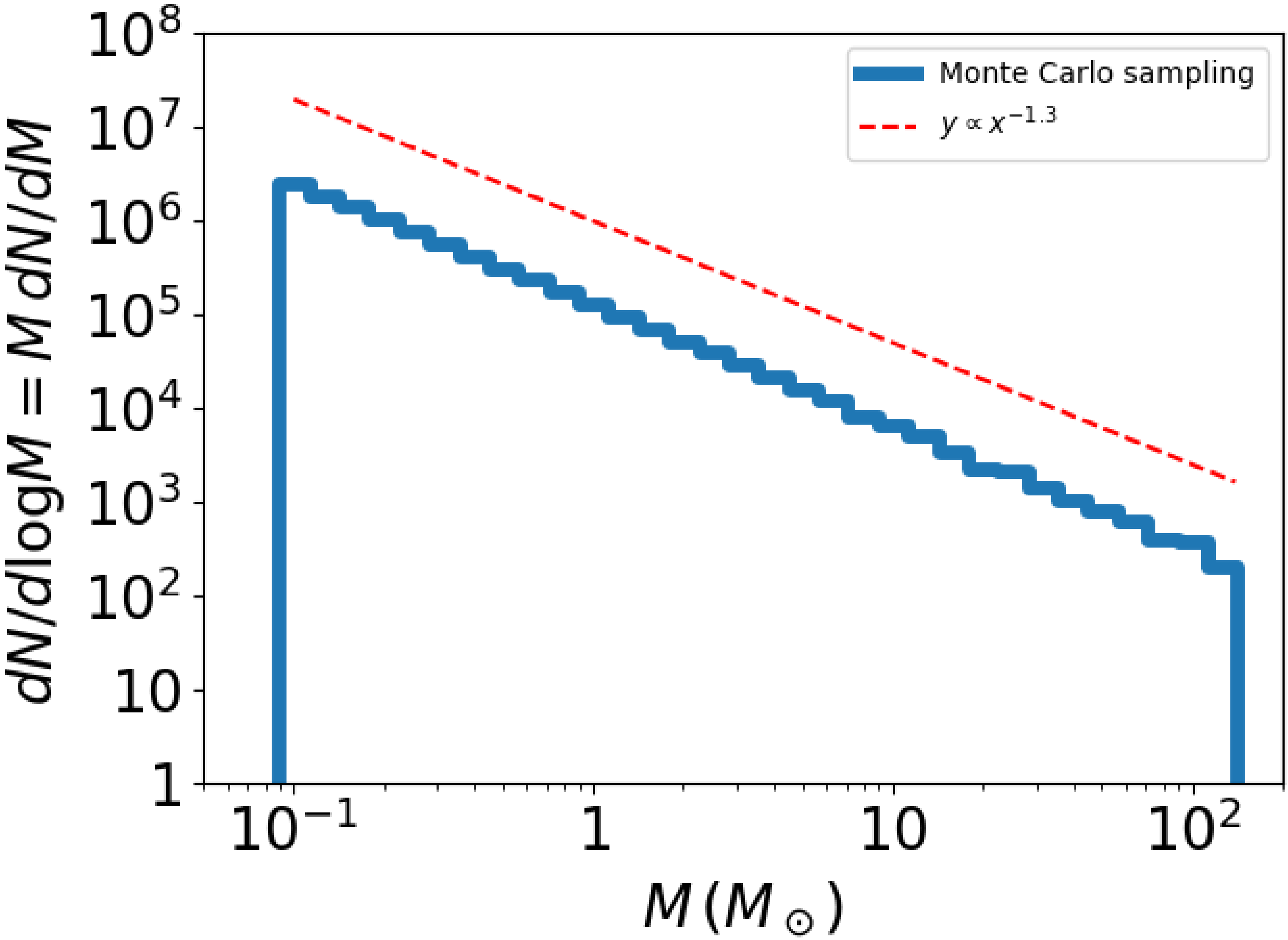
Random numbers. Inverse random sampling



If you do not use density = True but you want a logarithmic y-axis, you need to divide the values on the y axis by *M*, because

$$d\log M = dM/M \rightarrow dN/dM = M^{-1}\,dN/d\log M$$

Random numbers. Inverse random sampling



If you do not divide the y axis values by *M*, you get a slope as -1.3

Random numbers. Gaussian distribution with Box-Muller

Gaussian distribution: $$p(x) = \frac{1}{\sqrt{2\,\pi\sigma^2}}\,\exp\left(\frac{-x^2}{2\,\sigma^2}\right)$$

Cumulative probability distribution: cannot be inverted $$P(x) = \frac{1}{\sqrt{2\,\pi\sigma^2}}\int_{-\infty}^{x}\exp\left(\frac{-x'^2}{2\,\sigma^2}\right)\mathrm{d}x'$$

However, take the product of 2 Gaussians

$$p(x)\mathrm{d}x\,p(y)\mathrm{d}y = \frac{1}{\sqrt{2\,\pi\sigma^2}}\,\exp\left(\frac{-x^2}{2\,\sigma^2}\right)\mathrm{d}x\,\frac{1}{\sqrt{2\,\pi\sigma^2}}\,\exp\left(\frac{-y^2}{2\,\sigma^2}\right)\mathrm{d}y$$
$$= \frac{1}{2\,\pi\sigma^2}\,\exp\left(-\frac{x^2+y^2}{2\,\sigma^2}\right)\mathrm{d}x\,\mathrm{d}y$$

Let's use the transformation into polar coordinates: $$x^2+y^2 \to r^2$$
$$\mathrm{d}x\,\mathrm{d}y \to r\,\mathrm{d}r\,\mathrm{d}\theta$$

and convert the product of two Gaussians in polar coordinates

$$p(r,\theta)\,\mathrm{d}r\,\mathrm{d}\theta = \frac{1}{2\,\pi\sigma^2}\,\exp\left(-\frac{r^2}{2\,\sigma^2}\right)r\,\mathrm{d}r\,\mathrm{d}\theta$$
$$= \frac{r}{\sigma^2}\,\exp\left(-\frac{r^2}{2\,\sigma^2}\right)\mathrm{d}r\,\frac{\mathrm{d}\theta}{2\,\pi}$$

Random numbers. Gaussian distribution with Box-Muller

From the previous equation we can perfectly separate the terms in *r* and the terms in *θ* :

$$p(r)\,\mathrm{d}r = \frac{r}{\sigma^2}\,\exp\left(-\frac{r^2}{2\,\sigma^2}\right)\mathrm{d}r$$

$$p(\theta)\,\mathrm{d}\theta = \frac{1}{2\,\pi}\,\mathrm{d}\theta$$

These two separate functions can easily be integrated and inverted

$$P(r) = \int_0^r \frac{r'}{\sigma^2}\,\exp\left(-\frac{r'^2}{2\,\sigma^2}\right)\mathrm{d}r' = 1 - \exp\left(-\frac{r^2}{2\,\sigma^2}\right)$$

$$P(\theta) = \int_0^\theta \frac{1}{2\,\pi}\mathrm{d}\theta' = \frac{\theta}{2\,\pi}$$

$\longrightarrow$

$$r = \sqrt{-2\,\sigma^2\,\ln\left(1 - P(r)\right)}$$

$$\theta = 2\,\pi\,P(\theta)$$

We now use the inverse sampling method to generate two uniform random numbers z_1 and z_2 based on the previous equation

$$r = \sqrt{-2\,\sigma^2\,\ln\left(1 - z_1\right)}$$

$$\theta = 2\,\pi\,z_2$$

Random numbers. Gaussian distribution with Box-Muller

Finally, we derive x and y by using the transformation to polar coords

$$x = r\,cos(\theta)$$

$$y = r\,sin(\theta)$$

each of them distributed according to a Gaussian centered
in zero with standard deviation σ

– We can use one of the two numbers and store the other one for the future.

– If we want a Gaussian with a different mean value, we can simply shift the random numbers by the desired value.

Random numbers. Gaussian distribution with python

At least two functions in python to generate Gaussians:

numpy.random.normal(loc=0.0, scale=2.0)

where loc is the mean and scale the standard deviation

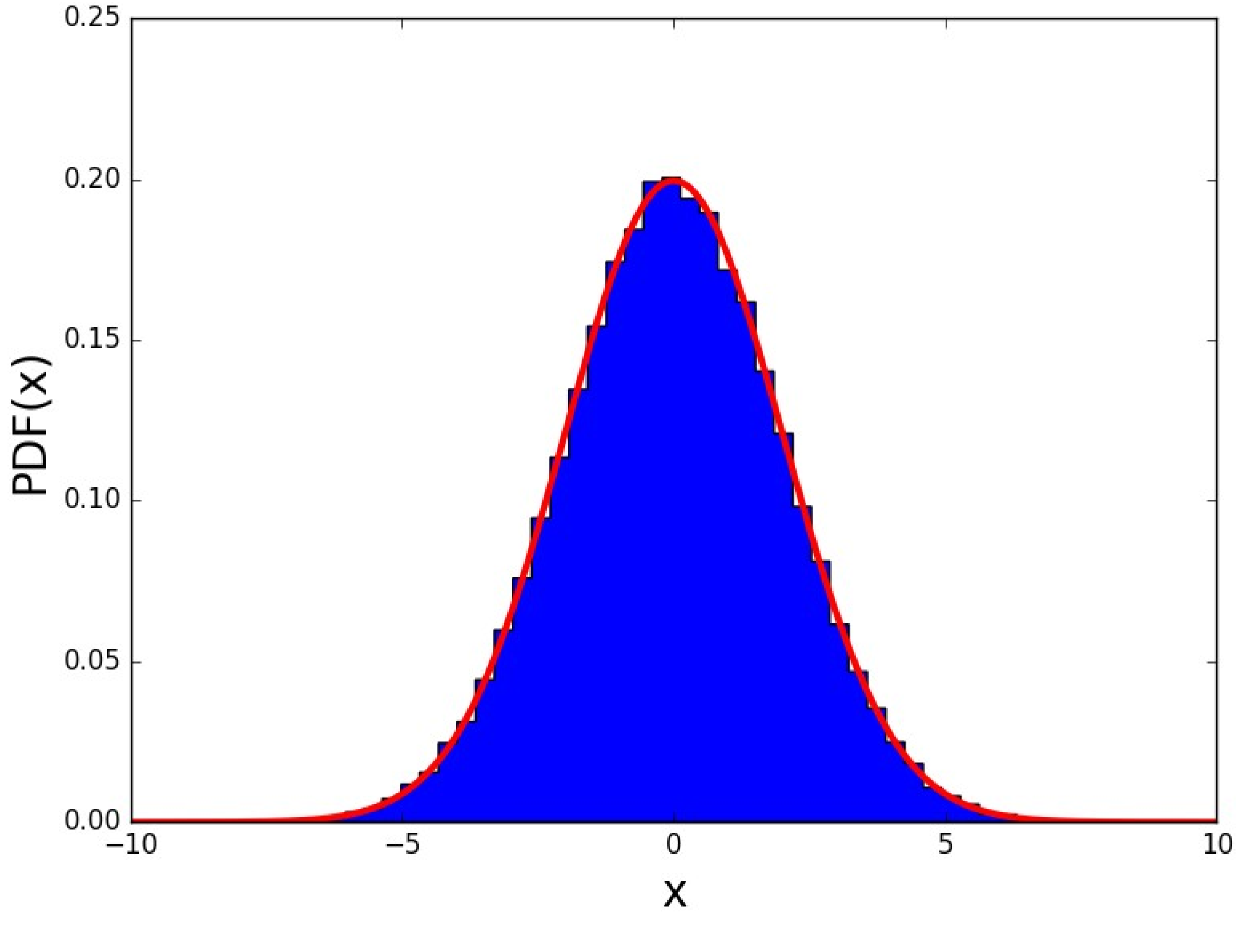
random.gauss(0.0, 2.0)

where first argument is the mean and
second argument is the standard deviation

Random numbers. Exercise, Gaussian with Box-Mueller

EXERCISE:

Write a script to generate $N = 10^5$ Gaussian deviates with the Box-Muller method. Assume $\sigma = 2$ and that the Gaussian is centered on zero. The result should look like Figure 31.



Random numbers. Rejection method

What can we do when the cumulative distribution function cannot be inverted (easily)? REJECTION SAMPLING APPROACH.

1. Take a probability distribution function p(x) of the quantity x you want to sample. But p(x) is difficult/impossible to integrate!

2. Take a second function *f* (x), with *f* (x) > p(x) everywhere, that can be easily integrated, to obtain the cumulative distribution function

$$g(x) = \int_{x_{\min}}^{x} f(x')\mathrm{d}x'$$

Note that g(x) is NOT a well defined probability.

3. Randomly sample y = g(x) between min and max value.

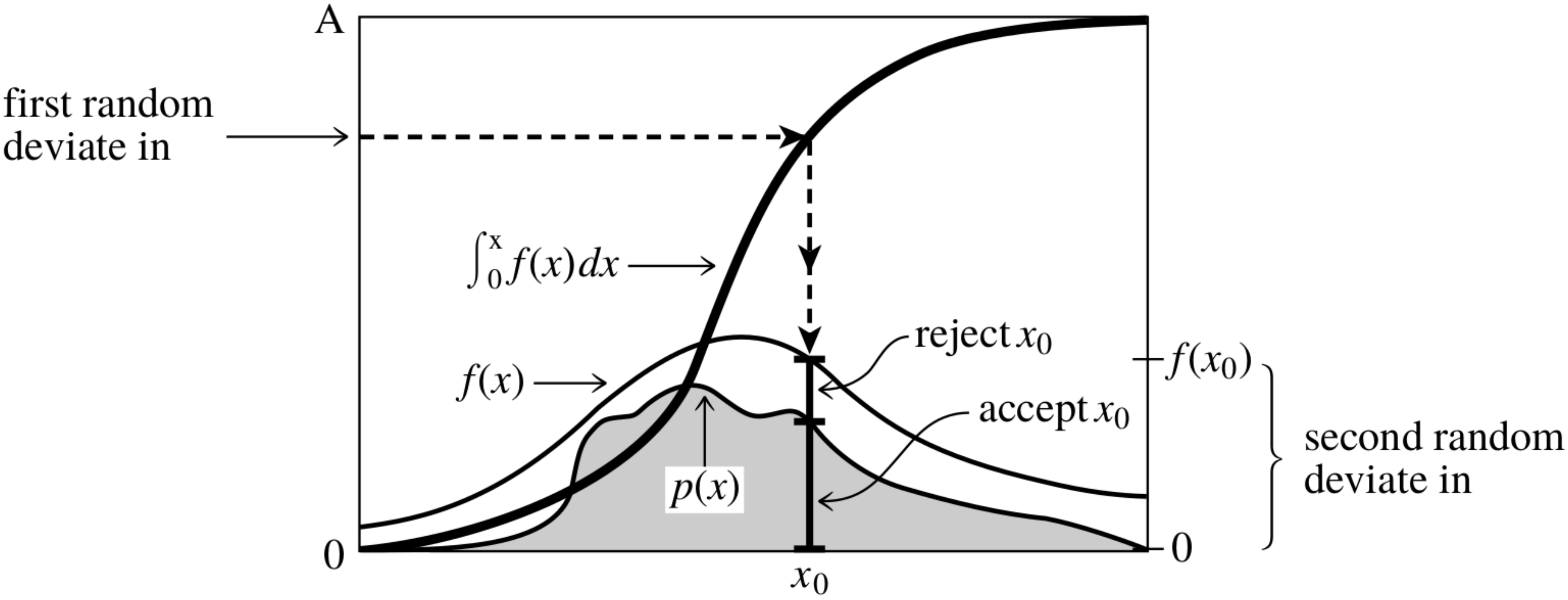
4. Invert g(x) to obtain x. x is distributed according to *f* (x).

5. Generate a second random number m uniform between 0 and *f* (x). Reject x if m > p(x) and accept x if m ≤ p(x).

6. Repeat 3, 4 and 5 as many times as you need to get x for N particles.

Random numbers. Rejection method

Better understood with a figure:



Random numbers. Exercise, Gaussian Rejection

EXERCISE:

Generate 10^4 points distributed according to a Gaussian (with $\sigma = 2$ and centered around zero) with the rejection method, by using the distribution function $f(x) = 1$, uniform between $min = -50$ and $max = +50$. The result should look like Figure 33.

Suggestion: once again, note that $f(x)$ is not a well defined probability distribution function – and it cannot be, because $f(x) > p(x)$ everywhere, where $p(x)$ is a well defined probability function (the Gaussian PDF in this case). Hence $y(x) = \int_{x_{\min}}^{x} f(x)\,dx$ is a uniform random number but does not necessarily lie in the interval between 0 and 1. You must first calculate

$$y_{\max} = \int_{x_{\min}}^{x_{\max}} f(x)\,dx \tag{90}$$

Hence, you should draw an uniform random number $y \in [0, y_{\max}]$.

Random numbers. Exercise, Gaussian Rejection

EXERCISE:

Generate 10^4 points distributed according to a Gaussian (with $\sigma = 2$ and centered around zero) with the rejection method, by using the distribution function $f(x) = 1$, uniform between $min = -50$ and $max = +50$. The result should look like Figure 33.

Suggestion: once again, no

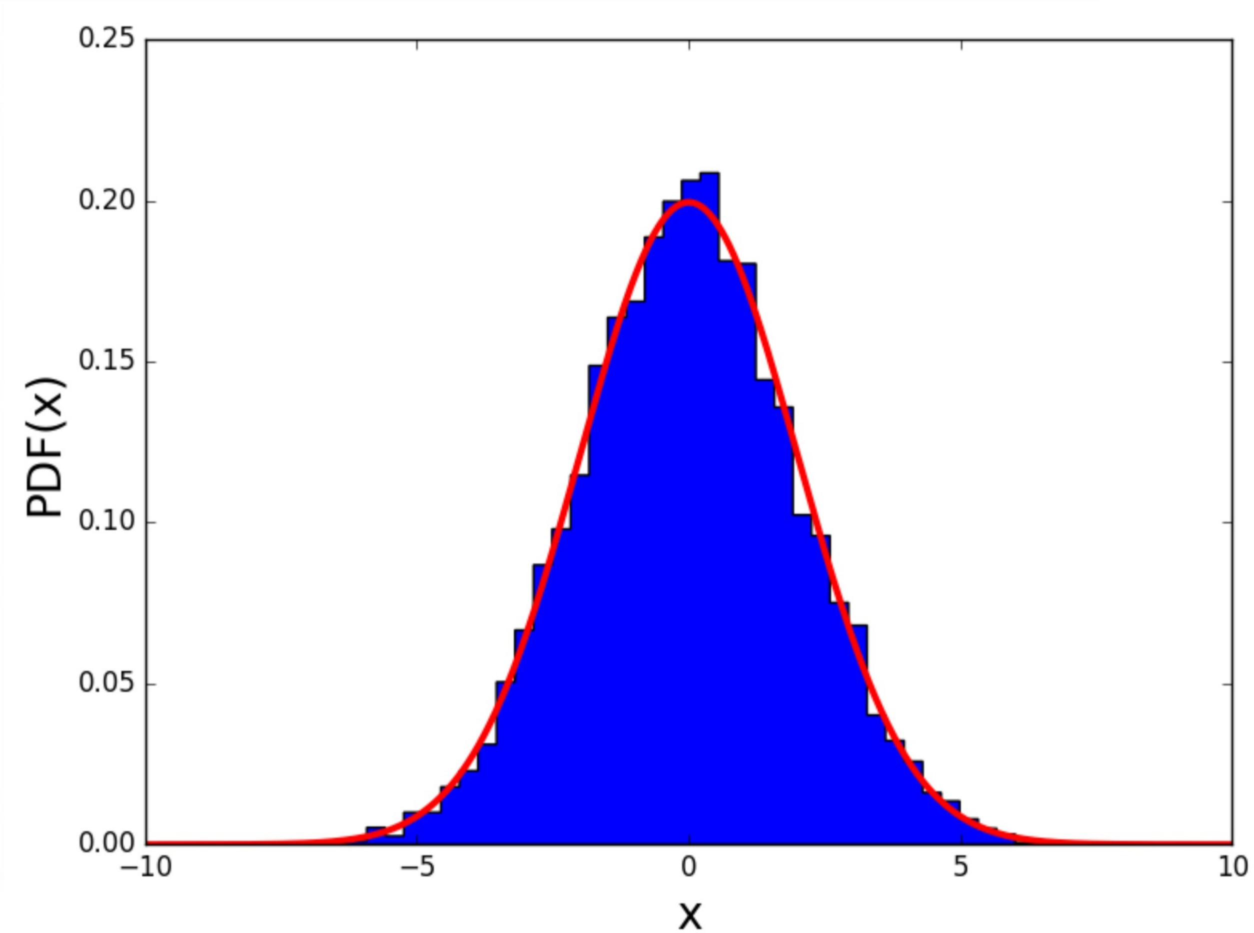
distribution function – and

where $p(x)$ is a well defined

case). Hence $y(x) = \int_{x_{\min}}^{x} f(x$

necessarily lie in the interva

Hence, you should draw an



Random numbers. Maxwellian from Gaussian

It can be shown that a Maxwellian curve $$p(v)\,\mathrm{d}v = \sqrt{\frac{2}{\pi}}\,\frac{v^2}{\sigma^3}\,\exp\left(\frac{-v^2}{2\,\sigma^2}\right)\mathrm{d}v$$

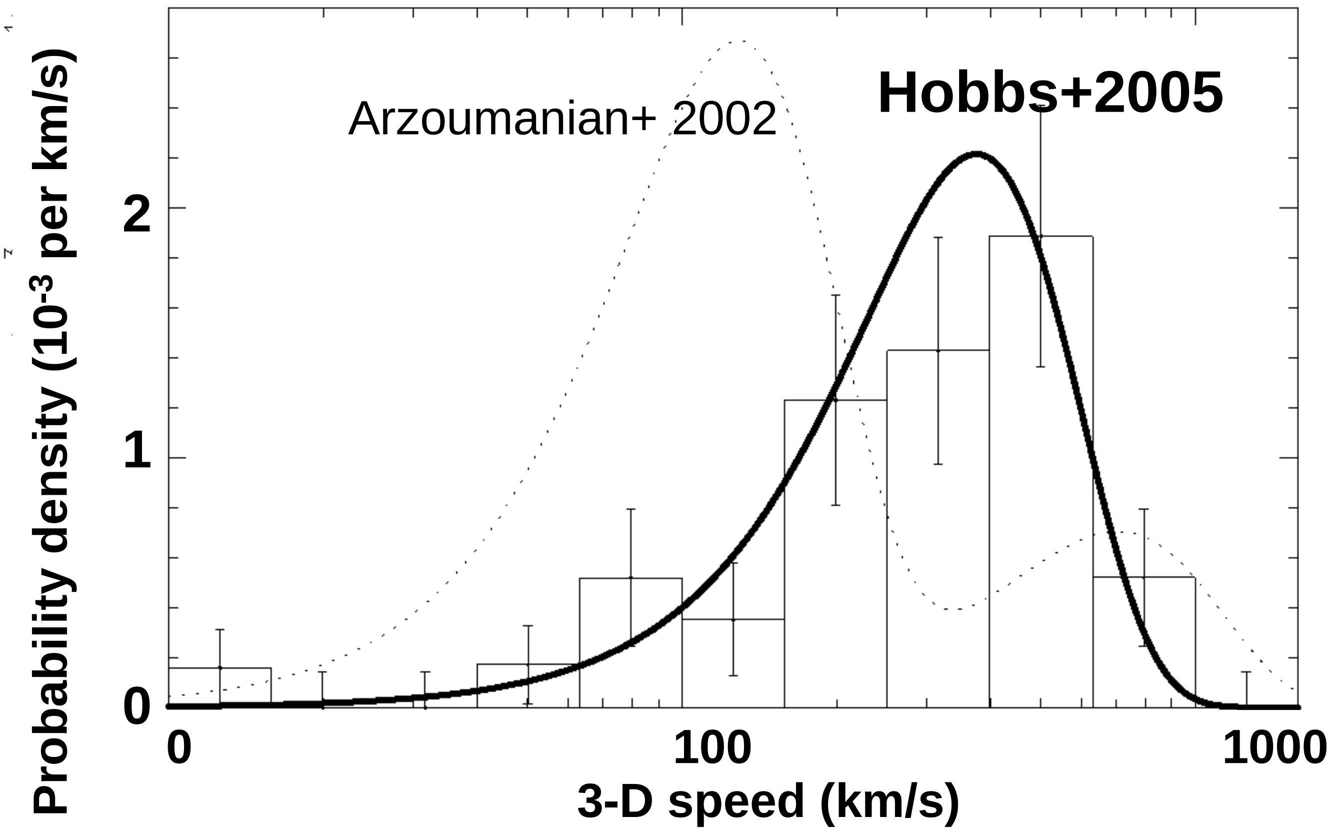
can be randomly sampled as $$v = \sqrt{(x^2 + y^2 + z^2)}$$

where *x, y, z* are Gaussian deviates centered around zero

A Maxwellian curve is a good representation for:

1. NATAL KICKS OF PULSARS

2. MOTIONS OF STARS IN A RELAXED AND VIRIALIZED SYSTEM

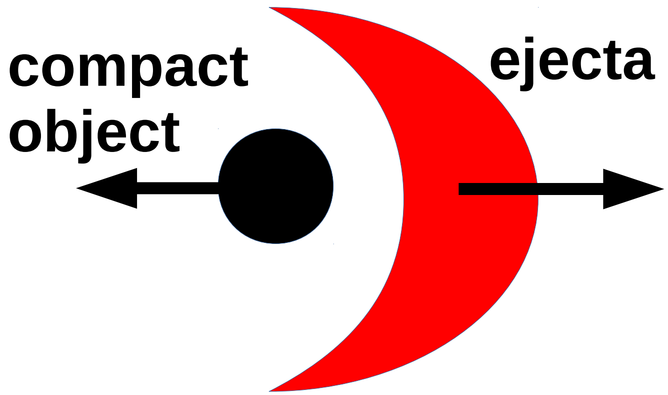


Random numbers. Maxwellian from Gaussian

EXERCISE:

Write a script to generate $N = 10^5$ random numbers following a Maxwellian distribution with $\sigma = 265$ km s^{-1}. According to Hobbs et al. [2005], this is the distribution of natal kicks of neutron stars.

Note: every compact object which forms from a supernova is thought to receive a kick at birth. The main reason is that linear momentum is conserved during a supernova and asymmetries in the ejecta (or in neutrino losses) push the compact object to move in the opposite direction with respect to the bulk of the ejecta.



Random numbers. Exercise "Star cluster"

Let's build an N-body model for a star cluster with N = 10^3 stars, where

- Stellar masses are drawn from a **Salpeter** mass function
- positions are randomly drawn from a **Plummer** sphere density distribution
- velocities are randomly drawn from a **Maxwellian** curve

For simplicity, let us assume that:

- the mass of a star does not depend on star's position inside the cluster (equivalent to no initial mass segregation);
- the velocity of a star does not depend on star's position inside the cluster;
- the distribution of stellar positions and velocities are isotropic.
- the cluster is in virial equilibrium (virial ratio 2 K/|W| = 1)

First step: draw the stellar masses from the Salpeter initial mass function (IMF), as in the previous exercise.

As you can simply verify, drawing 10^3 stars is equivalent to a mass of ~300 – 500 Msun (a part from stochastic fluctuations), because the average stellar mass is ~ 0.3 – 0.5 Msun.

Save the exact value of the total mass in M.

Random numbers. Exercise "Star cluster"

Second step: draw the stellar positions from a Plummer sphere

The Plummer sphere is a density distribution function expressed as

$$\rho(r) = \frac{3\,M}{4\,\pi\,a^3}\left(1 + \frac{r^2}{a^2}\right)^{-5/2}$$

where ρ is the mass density, a is a typical scale-length,

r is the radial coordinate (in spherical coordinates) and M is the total mass of the sphere (for M use the total mass that you generated with the Salpeter distribution).

Let us assume a = 1 pc.

The Plummer sphere is the simplest distribution function mimicking the radial distribution of stars in a star cluster.

Third step: draw stellar velocities from a Maxwellian distribution

$$f(v) = \sqrt{\frac{2}{\pi}}\,\frac{v^2\ \exp\left[-v^2/(2\,\sigma^2)\right]}{\sigma^3}$$

assume σ = 0.5 km s^-1 is the one-dimensional root-mean-square velocity of the Maxwellian curve. **Then velocities will be rescaled to enforce virial equilibrium**

Random numbers. Exercise "Star cluster"

Additional suggestions:

* Plummer is a density distribution, while we need the mass distribution assuming the cluster is isotropic, hence:

$$\mathrm{d}m = \rho\,\mathrm{d}V = \rho(r)\,r^2\,\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi\,\mathrm{d}r$$

where I have used the definition of the volume element in spherical coordinates

$$\mathrm{d}V = r^2\,\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi\,\mathrm{d}r$$

The good news of assuming spatial isotropy is that the angular coordinates are independent from each other and from the radial coordinate
→ we can draw r, θ and ϕ as three independent random numbers, from three different distributions.

Let's start with ϕ, which is the simplest one.
The cumulative probability distribution of ϕ is $P(\phi) = \frac{1}{2\,\pi}\int_0^{\phi}\mathrm{d}\phi' = \frac{\phi}{2\,\pi}$

Thus, we can draw a uniform random number $P(\phi)$ from 0 to 1 and then estimate ϕ as $\phi = 2\pi\, P(\phi)$

Random numbers. Exercise “Star cluster”

Now, let's calculate θ: The cumulative probability distribution of θ is

$$P(\theta) = \frac{1}{2}\int_0^{\theta} \sin\theta' \, \mathrm{d}\theta' = \frac{1}{2}\,(1-\cos\theta)$$

Thus, I can draw a uniform random number $P(\theta)$ between 0 and 1 and then I can extract θ from $P(\theta)$ by simply inverting the above equation:

$$\theta = \arccos\left[1 - 2\,P(\theta)\right]$$

To extract r is a bit more complicated. The cumulative distribution function of mass is

$$M(r) = \int_V \rho \, \mathrm{d}V = \int_0^{2\pi} \mathrm{d}\phi \int_0^{\pi} \sin\theta \, \mathrm{d}\theta \int_0^{r} \rho(r)\, r^2 \, \mathrm{d}r,$$

$$= 4\,\pi \int_0^{r} \frac{3\,M}{4\,\pi\,a^3}\left(1+\frac{r^2}{a^2}\right)^{-5/2} r^2\,\mathrm{d}r = M\left(\frac{r}{a}\right)^3 \left(1+\frac{r^2}{a^2}\right)^{-3/2}$$

Hence, the probability cumulative distribution function is $P(r) = \left(\frac{r}{a}\right)^3 \left(1+\frac{r^2}{a^2}\right)^{-3/2}$

After few math. steps, we find that equation can be inverted. We can draw a uniform random number $P(r)$ from 0 and 1 and then obtain

$$r = \sqrt{\frac{a^2}{P(r)^{-2/3} - 1}}$$

Random numbers. Exercise "Star cluster"

Finally, we randomly draw 10^3 values for *r*, θ and ϕ.
It might be easier to plot our points in Cartesian coordinates, thus we simply make the conversion:

$$x = r\,\sin\theta\,\cos\phi$$
$$y = r\,\sin\theta\,\sin\phi$$
$$z = r\,\cos\theta$$

Note that all direct N-body codes I am aware of work in Cartesian coordinates.

Now, we must assign a **velocity** to each of the 10^3 particles.

Since we assumed that stellar velocities do not depend on positions,
we do not need to worry about stellar position in the cluster (in real-life clusters we should).

Moreover, since we assumed that velocities are isotropic,
we can do the same as we did for positions and generate the modulus of velocity *v*, and the angles θ and ϕ independently of each other.
For θ and ϕ we do exactly as for the positions.

Random numbers. Exercise “Star cluster”

To generate *v* (modulus of the velocity), you can use the script you developed for the exercise on the **Box-Muller method** and extend it to generate Maxwellian deviates,

because it can be shown that $$v_{\mathrm{Maxwellian}} = \sqrt{v_x^2 + v_y^2 + v_z^2}$$

where *vx, vy* and *vz* are random deviates distributed according to a **Gaussian PDF** (with mean = 0 and with the same sigma)

Finally, once we have generated 10^3 new values for v, θ and ϕ, we can convert to Cartesian coordinates by using the following transformation

$$v_x = v \sin\theta \cos\phi$$
$$v_y = v \sin\theta \sin\phi$$
$$v_z = v \cos\theta$$

Note that these values of θ and ϕ must not be the same as the ones for positions (otherwise you generate a bias): just sample the new numbers from scratch.

Note that assuming a Maxwellian velocity distribution is not self-consistent with the choice of the Plummer, because the Plummer has its own energy distribution. Anyway, the Maxwellian makes the problem much easier to solve.

Random numbers. Exercise “Star cluster”

VIRIAL EQUILIBRIUM:

After you have generated the velocities, calculate KINETIC AND POTENTIAL ENERGY

Then estimate the VIRIAL RATIO Qvir:= 2 K / |W|

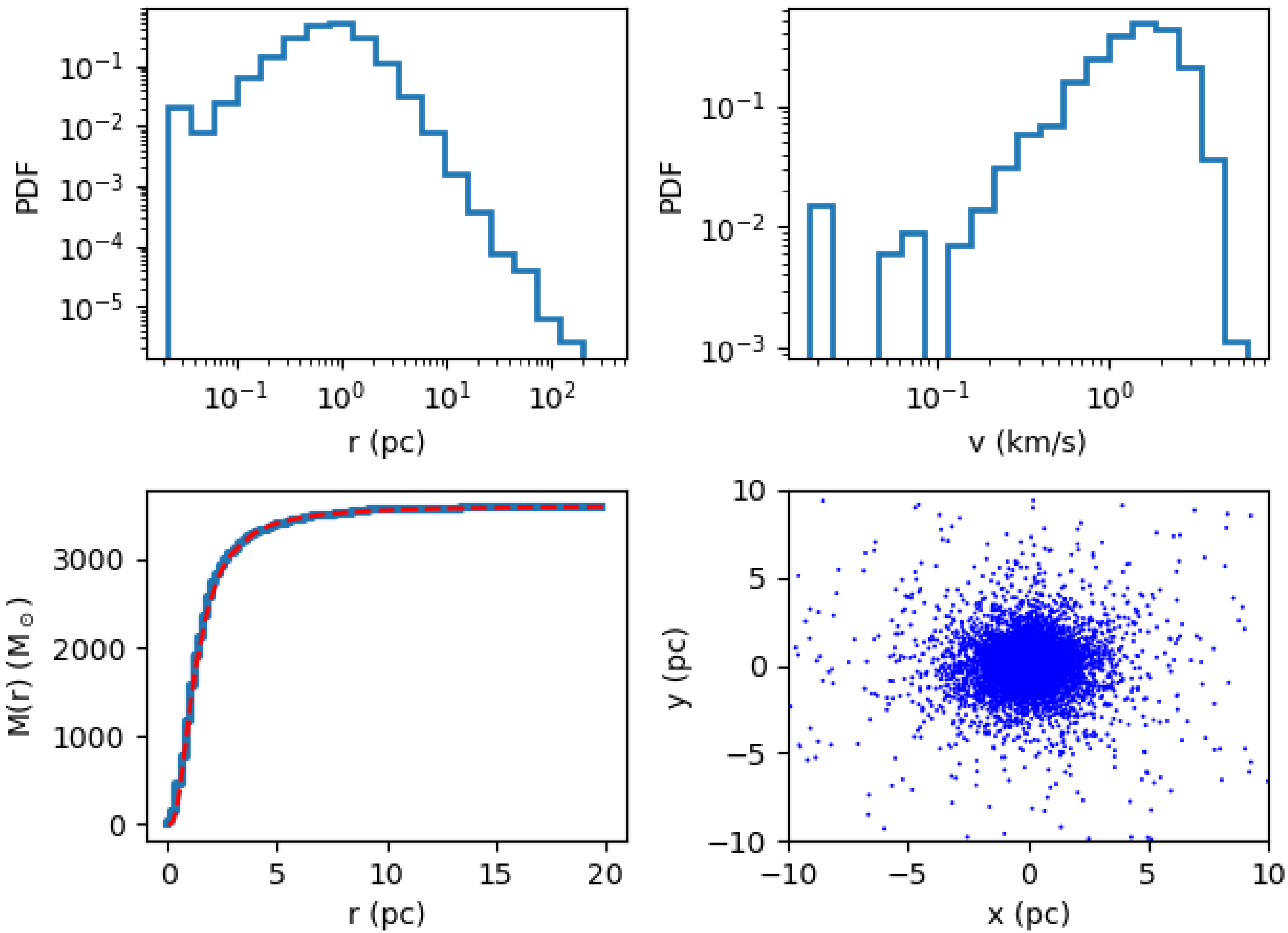
If Qvir != 1.0, the system is not in virial equilibrium

The fastest way to obtain a cluster in virial equilibrium is to divide all the components of the velocity by Qvir^0.5

$$v_{x,i} = v_{x,i}/\sqrt{Q_{vir}}$$
$$v_{y,i} = v_{y,i}/\sqrt{Q_{vir}}$$
$$v_{z,i} = v_{z,i}/\sqrt{Q_{vir}}$$

Random numbers. Exercise "Star cluster"

Result for a cluster with 10^4 stars and with α = 1 pc



examples/random/plummer.py