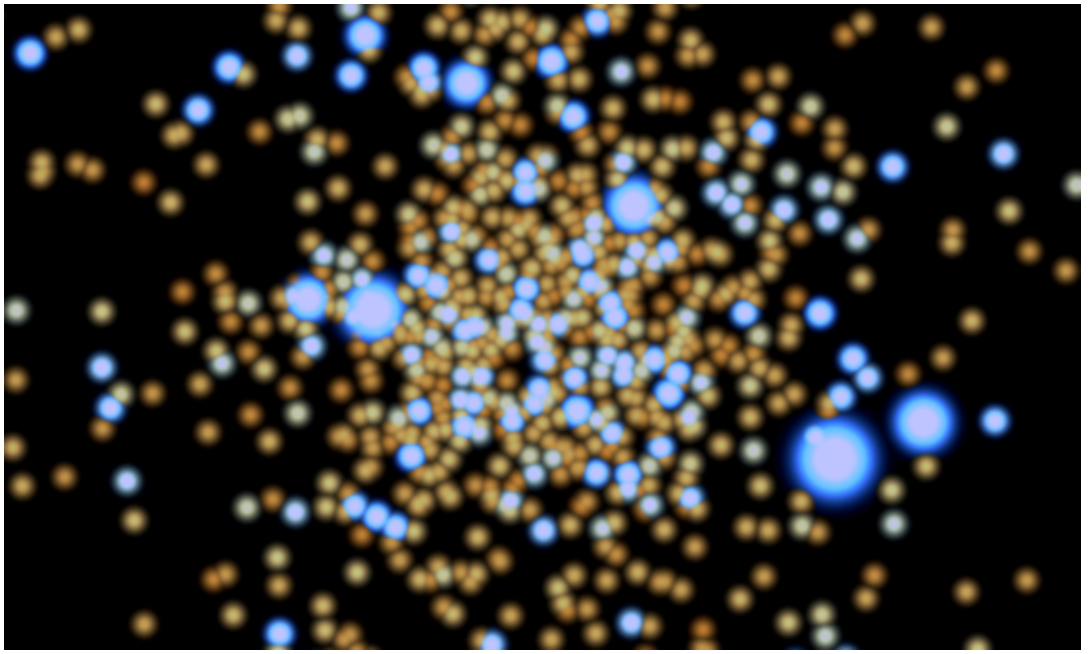
Michela Mapelli

N-body techniques for astrophysics: Lecture 2 – Monte Carlo techniques and Initial conditions

PhD School in Astrophysics,
University of Padova
November 19-30, 2018

The initial conditions of an N-body simulations:

- a system described as N particles



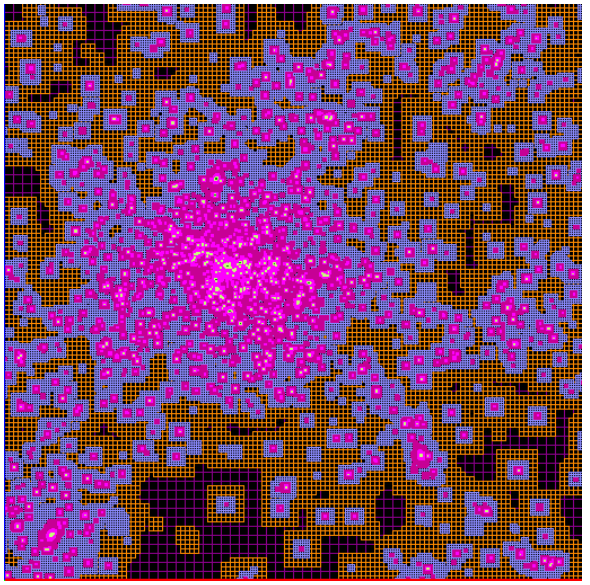
- I must assign positions, velocities, masses (and in some cases also other quantities – e.g. pressure, temperature, density, radius, softening, luminosity, metallicity, …) to each particle

HOW CAN I DO THIS?

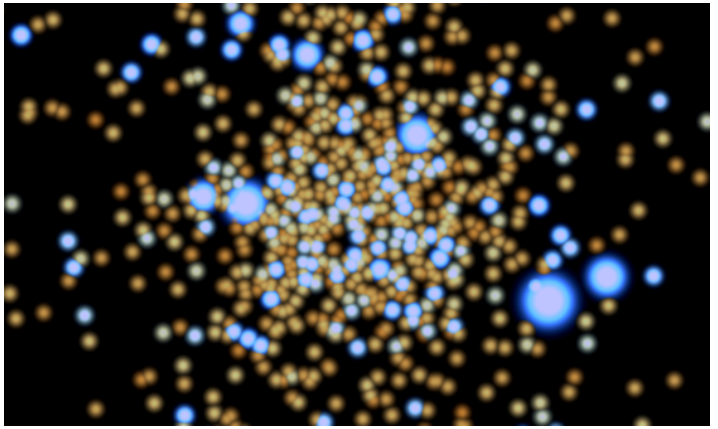
Distribution functions:

1. I need to know the best distribution function of my particles (for positions, velocities, masses, etc)

2a. If simulation is made of cells:
I sample the distribution function on a regular grid
→ The number of cells is my resolution



2b. If simulation is made of particles:
I SAMPLE the distribution function with Monte Carlo technique (e.g. I generate particles RANDOMLY as to be representative of the distribution function)



Monte Carlo technique:

IDEA: ask the computer to random generate a number
(approximately: randomly pick up)

Simplest case: random generate a number between 0 and Nmax
with the same probability of occurring in 0, Nmax
(UNIFORM DEVIATE from 0 to Nmax)

Requirements:

- **“genuine” randomness**
- **reproducibility**

How to do this?

DIFFICULT – see eg. Numerical Recipes book
https://en.wikipedia.org/wiki/Numerical_Recipes
Chapter 7

Monte Carlo technique:

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

where I_0 = seed **(same seed returns always same series - reproducibility)**

problem: after *m* iterations, the sequence begins identical!

E.g. this is how rand() works in c

Usage of rand:

```
void srand(unsigned seed);
int rand(void);

int main(){
      srand(199);
      int prova=rand();
      printf("%d\n", prova);
}
```

*** srand reads the user provided seed (199)**

*** rand generates integer uniformly distributed between 0 and Nmax**

*** check how big is RAND_MAX on your pc!**

Monte Carlo technique:

Better random generator: ran2 (example from Num. Recipes):

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0-EPS)

float ran2(long *idum)
```

Long period ($> 2 \times 10^{18}$) random number generator of L'Ecuyer with Bays-Durham shuffle and added safeguards. Returns a uniform random deviate between 0.0 and 1.0 (exclusive of the endpoint values). Call with idum a negative integer to initialize; thereafter, do not alter idum between successive deviates in a sequence. RNMX should approximate the largest floating value that is less than 1.

```
{
    int j;
    long k;
    static long idum2=123456789;
    static long iy=0;
    static long iv[NTAB];
    float temp;

    if (*idum <= 0) {                       Initialize.
        if (-(*idum) < 1) *idum=1;          Be sure to prevent idum = 0.
        else *idum = -(*idum);
        idum2=(*idum);
        for (j=NTAB+7;j>=0;j--) {           Load the shuffle table (after 8 warm-ups).
            k=(*idum)/IQ1;
            *idum=IA1*(*idum-k*IQ1)-k*IR1;
            if (*idum < 0) *idum += IM1;
            if (j < NTAB) iv[j] = *idum;
        }
        iy=iv[0];
    }
    k=(*idum)/IQ1;                          Start here when not initializing.
    *idum=IA1*(*idum-k*IQ1)-k*IR1;          Compute idum=(IA1*idum) % IM1 without
    if (*idum < 0) *idum += IM1;                overflows by Schrage's method.
    k=idum2/IQ2;
    idum2=IA2*(idum2-k*IQ2)-k*IR2;          Compute idum2=(IA2*idum) % IM2 likewise.
    if (idum2 < 0) idum2 += IM2;
    j=iy/NDIV;                              Will be in the range 0..NTAB-1.
    iy=iv[j]-idum2;                         Here idum is shuffled, idum and idum2 are
    iv[j] = *idum;                              combined to generate output.
    if (iy < 1) iy += IMM1;
    if ((temp=AM*iy) > RNMX) return RNMX;   Because users don't expect endpoint values.
    else return temp;
}
```

Monte Carlo technique:

And if I want to generate random numbers distributed according to a given distribution function (e.g. Gaussian)?

1. INVERSE TRANSFORM SAMPLING

2. REJECTION METHOD

SAMPLING TECHNIQUES: Inverse transform sampling

The probability p(x) dx of generating a number uniformly distributed between x and x+dx is given by

$$p(x)dx = \begin{cases} dx & 0 < x < 1 \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{(1)}$$

The probability distribution p(x) is normalized so that

$$\int_{-\infty}^{\infty} p(x)dx = 1$$

If we generate a uniform deviate x and then take a function y(x), the probability distribution p(y) dy is determined by the fundamental transformation law of probabilities which is

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

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

from (1) → **p(y) dy = dx**

It is always possible to generate a non-uniform random deviate from a uniform random deviate

SAMPLING TECHNIQUES: Inverse transform sampling

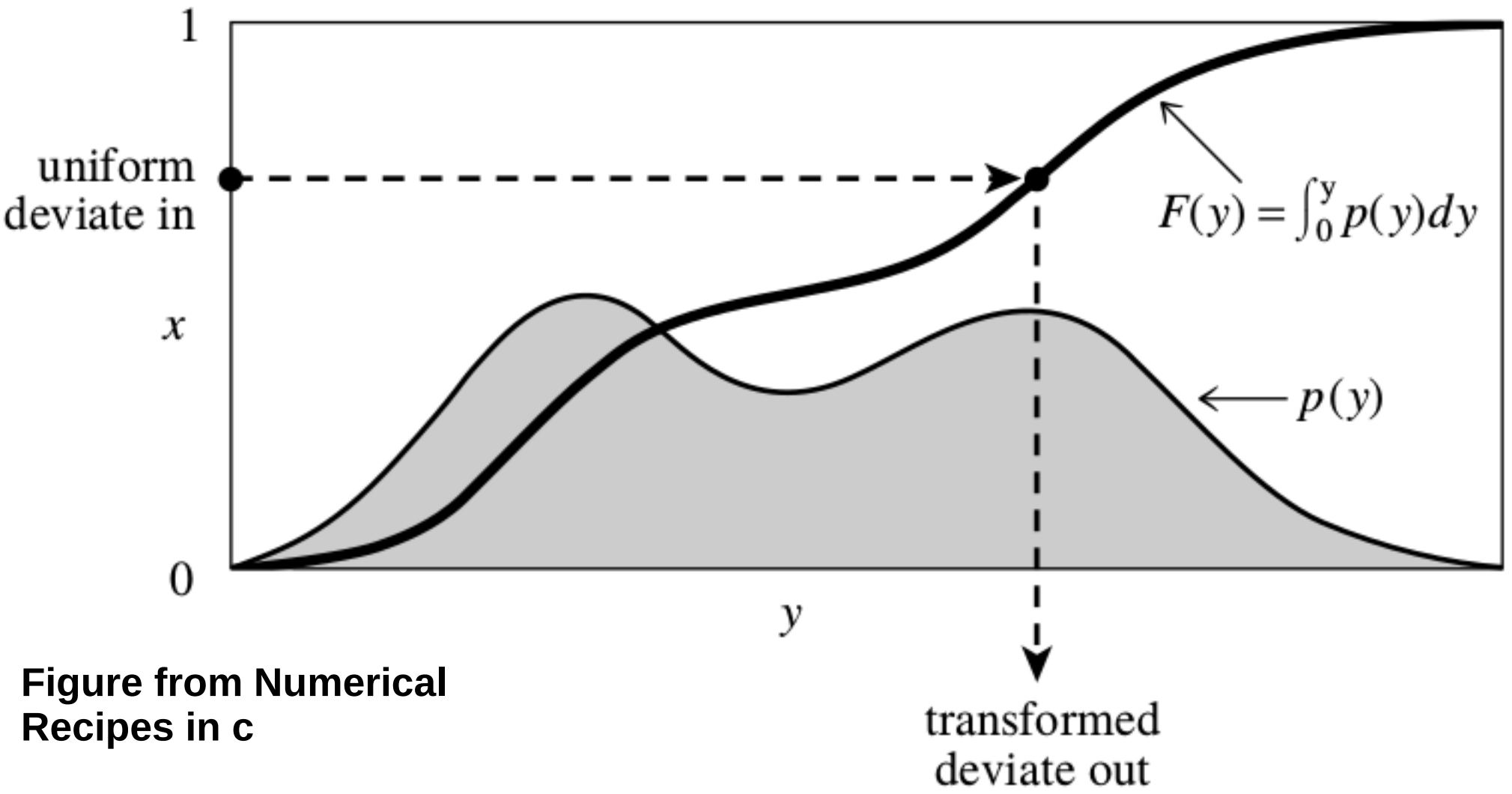


Figure from Numerical Recipes in c

If I generate x uniformly between 0 and 1, then I can obtain y(x) by inverting F(y)

SAMPLING TECHNIQUES: Inverse transform sampling

Inverse transform sampling (only for invertible distribution functions):

1- Take a probability distribution function *f(y)* of the quantity *y* I want to sample

2- Integrate it over the range to obtain the cumulative distribution function

$$F(y) = \int f(y)\, dy$$

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

3- Randomly sample the values *x = F(y)* of the cumulative distribution function between min and max value

4- Invert the function to get *y* *y = F(y)^-1*

REPEAT 3 & 4 as many times as you need to get *y* for N particles

SEE NUMERICAL RECIPES for more details!
http://www2.units.it/ipl/students_area/imm2/files/Numerical_Recipes.pdf

SAMPLING TECHNIQUES: Inverse transform sampling

Example: ECCENTRICITY *e*

1- *f(e) = 2 e*

called 'thermal distribution of eccentricity' (common in binary systems close to the Sun)

2- cumulative distribution function

$$F(e) = e^2\_max - e^2\_min$$

3- If e^2_max = 1 and e^2_min = 0, use a random generator to generate a random number between 0 and 1

$$e^2\_ran = \text{rand}(e^2\_max, e^2\_min)$$

4- Invert the function:

$$e\_ran = \text{sqrt}(e^2\_ran)$$

REPEAT 3 & 4 as many times as you need to get *e_ran* for N particles

SAMPLING TECHNIQUES: Inverse transform sampling

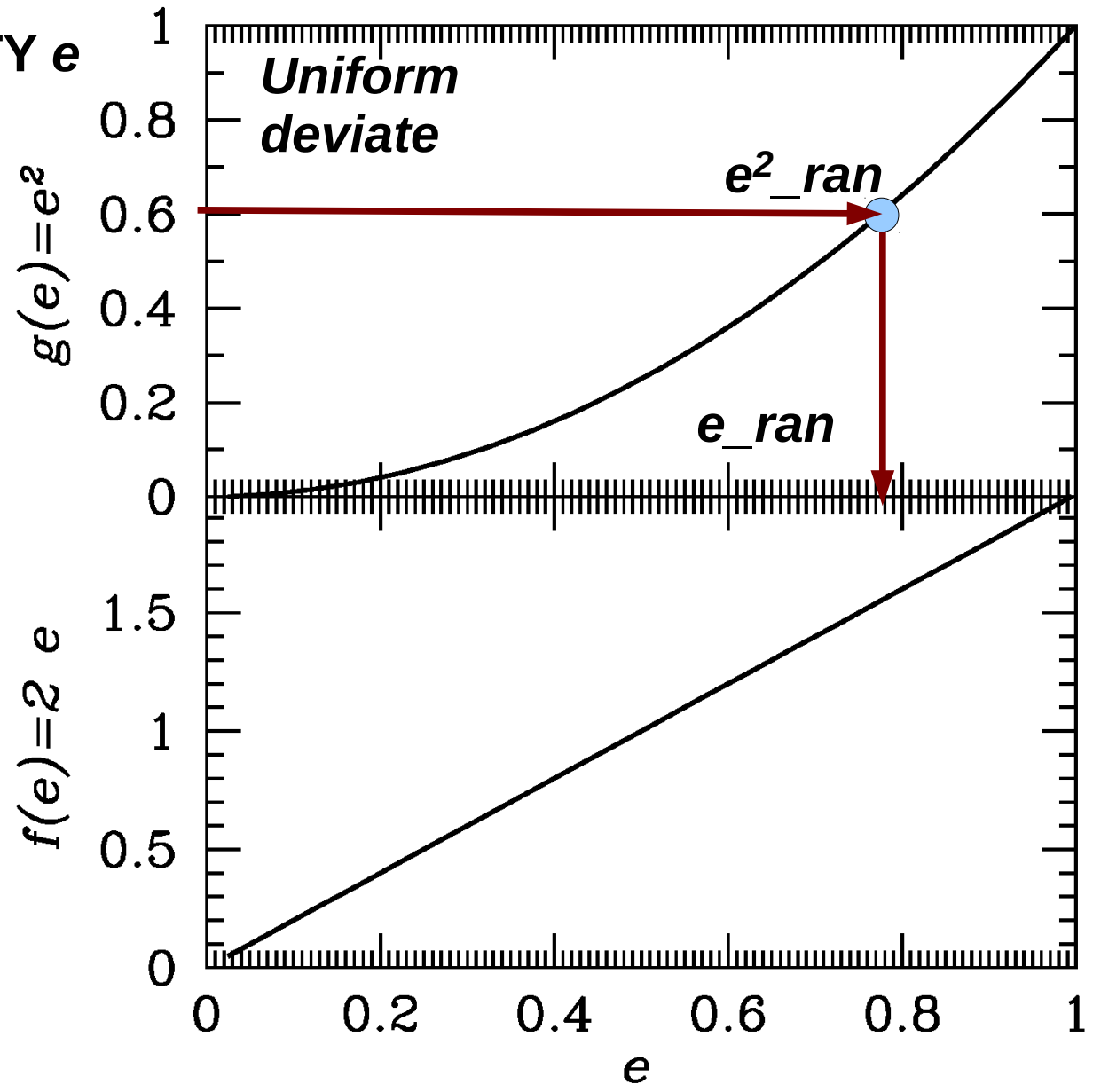
Example: ECCENTRICITY *e*

1- DF
f(e) = 2 e

2- cumulative DF
g(e) = e²

3-
***e²_ran* = rand()**

4- Invert the function:
***e_ran* = sqrt(*e²_ran*)**



SAMPLING TECHNIQUES: Inverse transform sampling

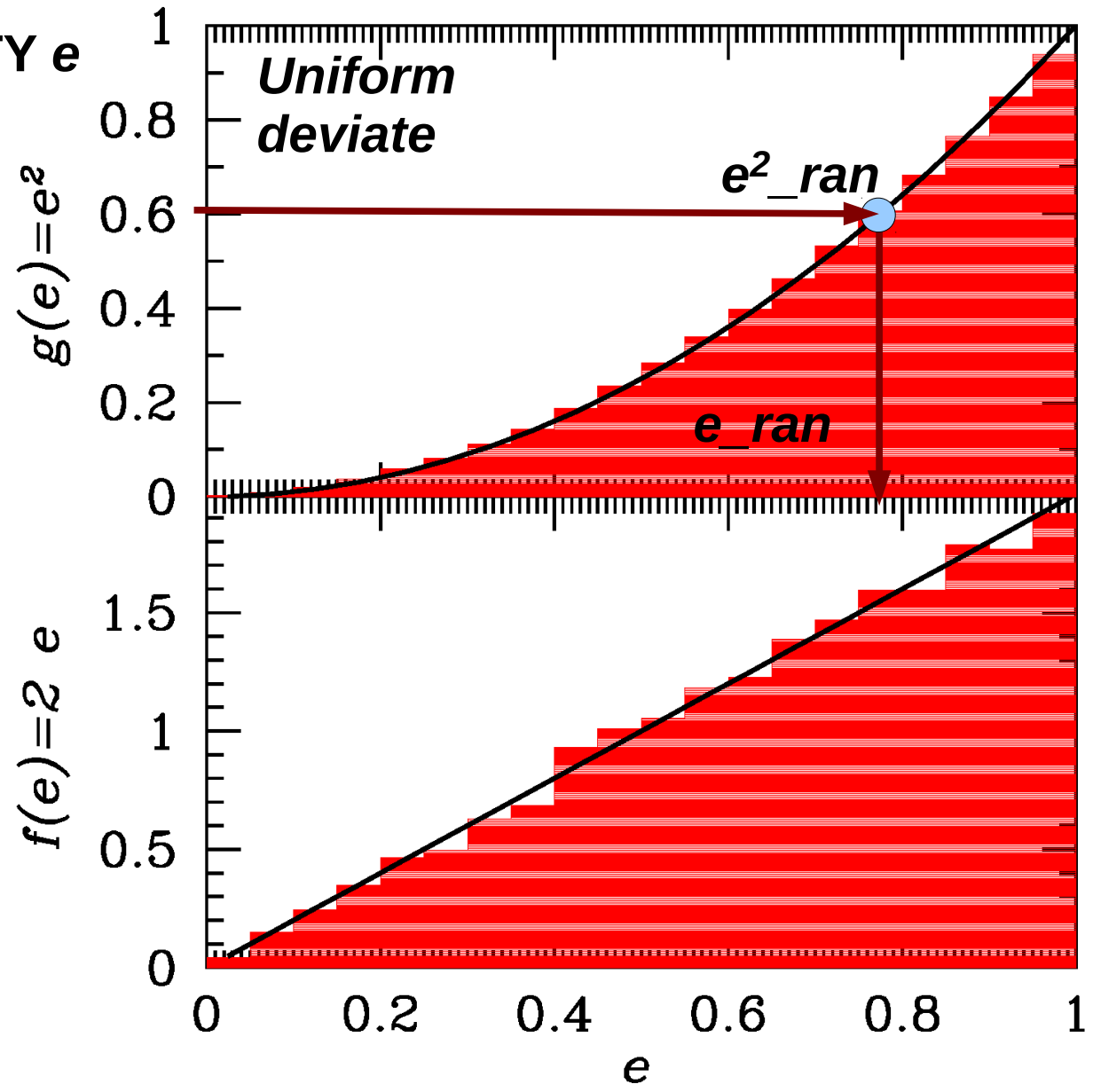
Example: ECCENTRICITY *e*

1- DF
f(e) = 2 e

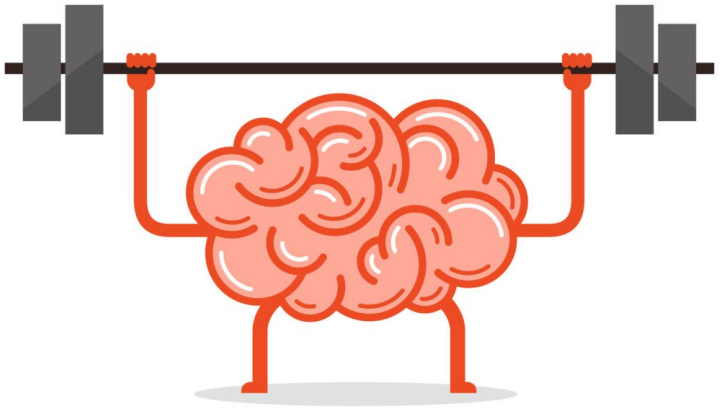
2- cumulative DF
g(e) = e²

3-
***e²_ran* = rand()**

4- Invert the function:
***e_ran* = sqrt(*e²_ran*)**



Exercise # 4:

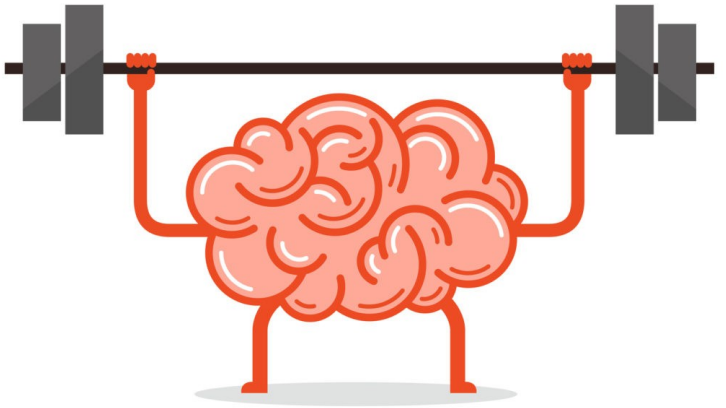


4.1 Generate 10’000 uniform deviates between 0.0 and 1
plot them

4.2 Generate 10’000 random numbers following an exponential distribution
$$p(y) = \exp(-y)\, dy$$
plot them

4.3 Generate 10’000 random numbers following a mass function
$$p(m) = m^{-a}\, dm$$
plot them

Exercise # 5:



Generate 10’000 points uniform inside the volume of a sphere of radius R

Note:
sphere volume element in spherical coordinates = r^2 dr sinθ dθ dϕ

The Gaussian curve with inverse transform sampling

The transform method can be generalized to multiple dimensions

$$p(y1, y2, ...)\, dy1\, dy2 ... = p(x1, x2, ...) \left| \frac{d(x1, x2, ...)}{d(y1, y2, ...)} \right| dy1\, dy2 ...$$

Jacobian determinant

The Gaussian function $p(y)\, dy = \frac{1}{\sqrt{2\,\pi}} \exp\left(-y^2/2\right) dy$

cannot be inverted, but we can use the Box-Muller method

Consider

$$y1 = \sqrt{-2\, \ln x1}\, \cos\left(2\,\pi\, x2\right)$$

$$y2 = \sqrt{-2\, \ln x1}\, \sin\left(2\,\pi\, x2\right)$$

Thus

$$x1 = \exp\left[-\frac{1}{2}\left(y1^2 + y2^2\right)\right]$$

$$x2 = \frac{1}{2\,\pi} \arctan\frac{y2}{y1}$$

The Gaussian curve with inverse transform sampling

The Jacobian determinant of this transformation is

$$\frac{\partial(x_1, x_2)}{\partial(y_1, y_2)} = \begin{vmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{vmatrix} = -\left[\frac{1}{\sqrt{2\pi}} e^{-y_1^2/2}\right]\left[\frac{1}{\sqrt{2\pi}} e^{-y_2^2/2}\right]$$

i.e. both y1 and y2 follow Gaussian distribution!

The Maxwellian curve with inverse transform sampling

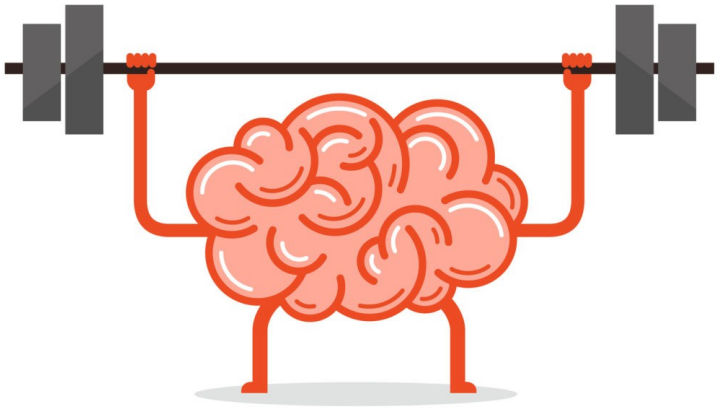
It can be shown that a Maxwellian curve $\sqrt{\frac{2}{\pi}} \frac{y^2 \exp -y^2/(2\,\sigma^2)}{\sigma^3}$

can be randomly sampled as

$$v = \sqrt{(x^2 + y^2 + z^2)}$$

Where *x*, *y* and *z* are three random numbers extracted from a Gaussian distribution with the same sigma

Exercise # 6:



6.1 Generate 10’000 points following Gaussian distribution with sigma=265 km/s

6.2 Generate 10’000 points following Maxwellian distribution with sigma=265 km/s

SAMPLING TECHNIQUES: Inverse transform sampling

AND WHAT DO I DO IF THE CUMULATIVE FUNCTION CANNOT BE INVERTED (EASILY)?

SAMPLING TECHNIQUES: Rejection sampling

Rejection sampling

1- Take a prob. distr. function *p(y)* of the quantity *y* I want to sample
But *p(y)* is difficult/impossible to integrate!

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

$$g(y) = \int f(y)\, dy$$

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

4- *Invert g(y) to obtain y*
y* is distributed according to *f(y)

5*- Generate a second random number *m* uniform between 0 and *f(y)
Reject *y* if *m>p(y)* and accept *x* if *m<=p(y)*

REPEAT 4 & 5 as many times as you need to get x for N particles

SAMPLING TECHNIQUES: Rejection sampling

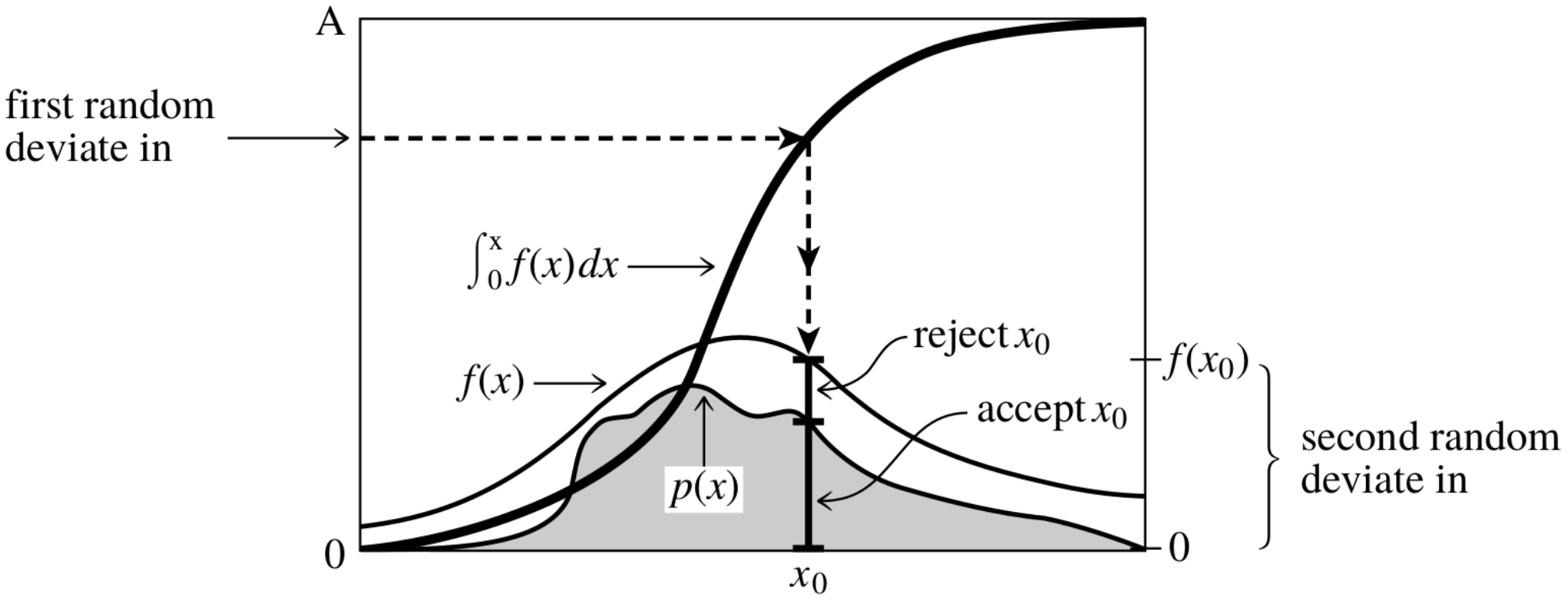
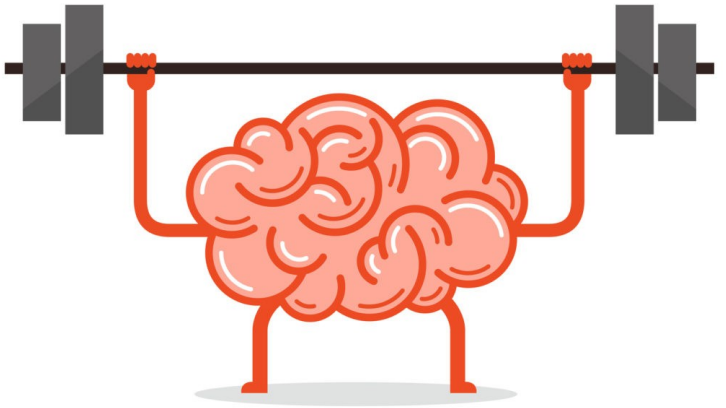


Figure from Numerical Recipes in c

Exercise # 7:



Generate 10’000 points uniform inside the volume of a sphere of radius R = 3 WITH REJECTION METHOD

Note:
sphere volume element in spherical coordinates = r^2 dr sinθ dθ dϕ

But do you really need this?

A more complicate example: draw particle positions from a Plummer sphere

A Plummer sphere (Plummer 1911):

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

2. what is the probability we should draw the random #s from?

$$dm = \rho(r)\,r^2\,\sin\theta\,d\theta\,d\phi\,dr$$

3. cumulative distribution function (mass):

$$M(r) = M\,\frac{r^3}{(r^2+a^2)^{3/2}}$$

Normalized to total mass: $$P(r) = \frac{r^3}{(r^2+a^2)^{3/2}}$$

A more complicate example: draw particle positions from a Plummer sphere

4. draw uniform random numbers to get *P(r)* $P(r) = \frac{r^3}{(r^2 + a^2)^{3/2}}$

5. invert *P(r)* to get *r*, or use rejection or use EXTRAPOLATION (*)

6. repeat it for N particles

7. random generate θ (homogeneous in cosθ) and *ϕ* (homogeneous in 2 π)

8. derive Cartesian coordinates as

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

A more complicate example: draw particle positions from a Plummer sphere

(*) EXTRAPOLATION is the simplest way to sample

1. prepare an array of *r* between 0 and R

2. calculate *P(r)* for each of these *r*

$$P(r) = \frac{r^3}{(r^2 + a^2)^{3/2}}$$

3. draw a random number *X* from 0 to 1

4. interpolate the array of *P(r)* to get *r* such that *X=P(r)*

A more complicate example: draw particle velocities from a Plummer sphere

Velocities are distributed approximately as a Maxwellian curve with σ *(r)*

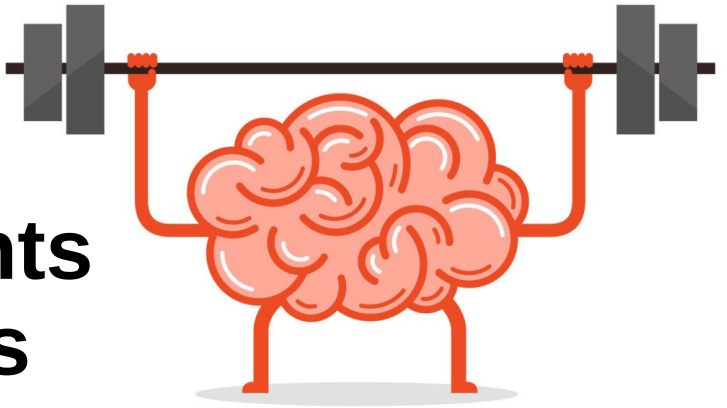
$$\sigma^2(r) = \frac{G\,M}{6\,(r^2 + a^2)^{1/2}}$$

Thus the probability of having a velocity *v* is

$$P(v, r) = 4\,\pi \int_0^v \left(\frac{1}{2\,\pi\,\sigma^2}\right)\, v^2 \exp\left(-\frac{v^2}{2\,\sigma^2}\right) dv$$

Uniform random numbers are drawn from this distribution and then inverted.

Exercise # 8:



Generate 1’000 random points with positions and velocities according to a Plummer sphere

Exercise # 8b:

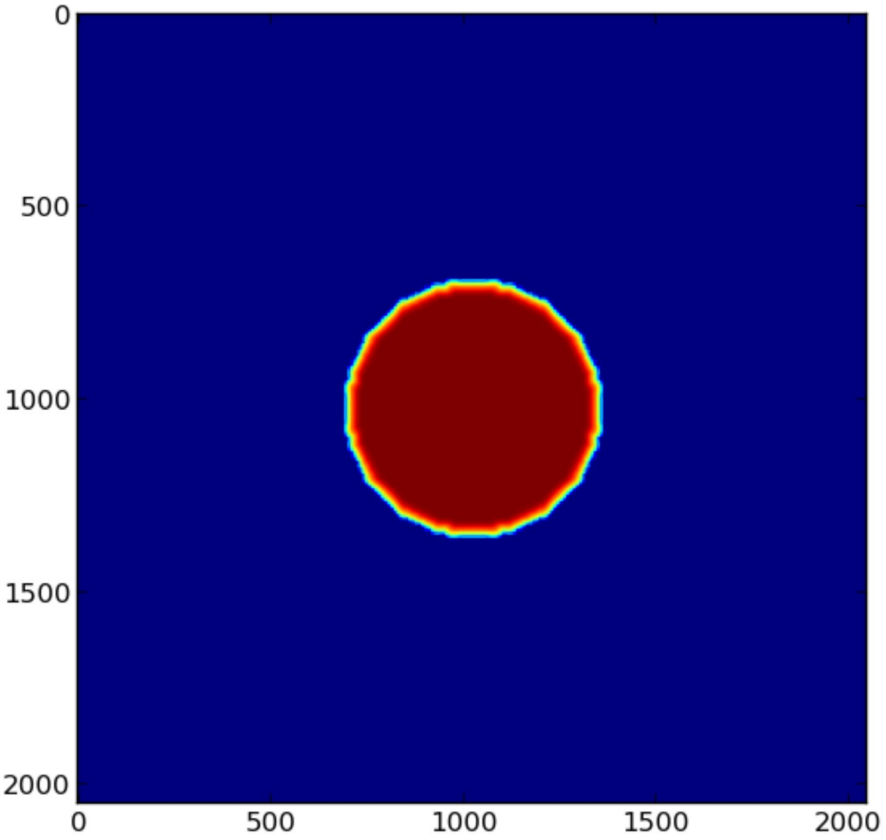
Simulate this system with the leapfrog code You produced for exercise 3b

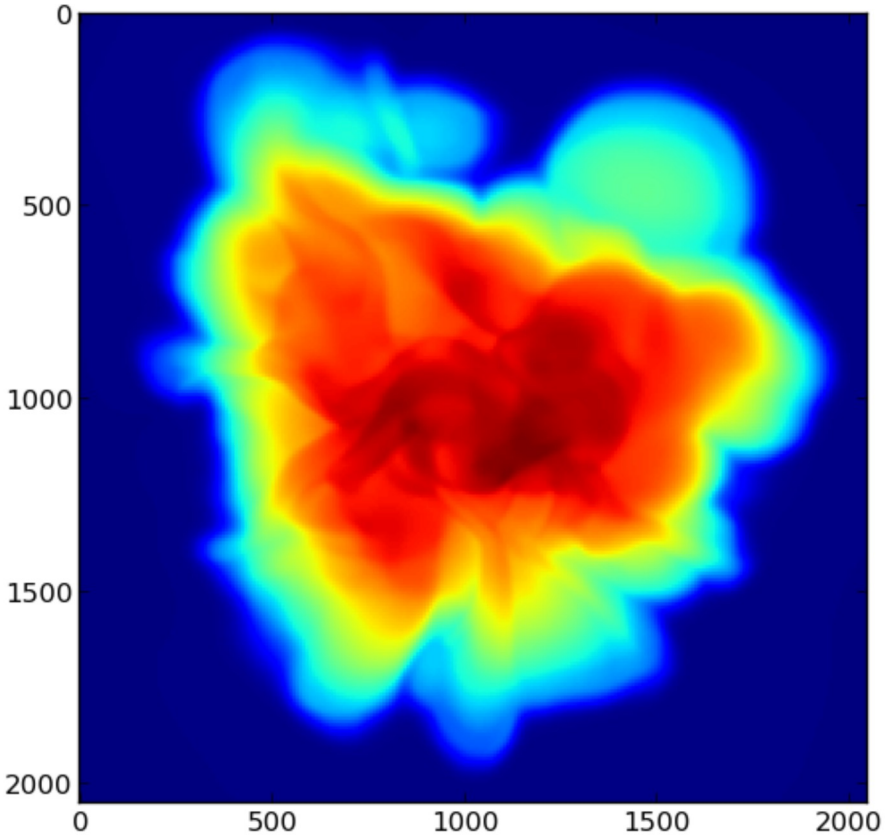
TO GENERATE A MOLECULAR CLOUD:

Uniform sphere of gas;

Total energy = 0 + Gaussian random motions;

Optional: seeded with turbulence as power spectrum.



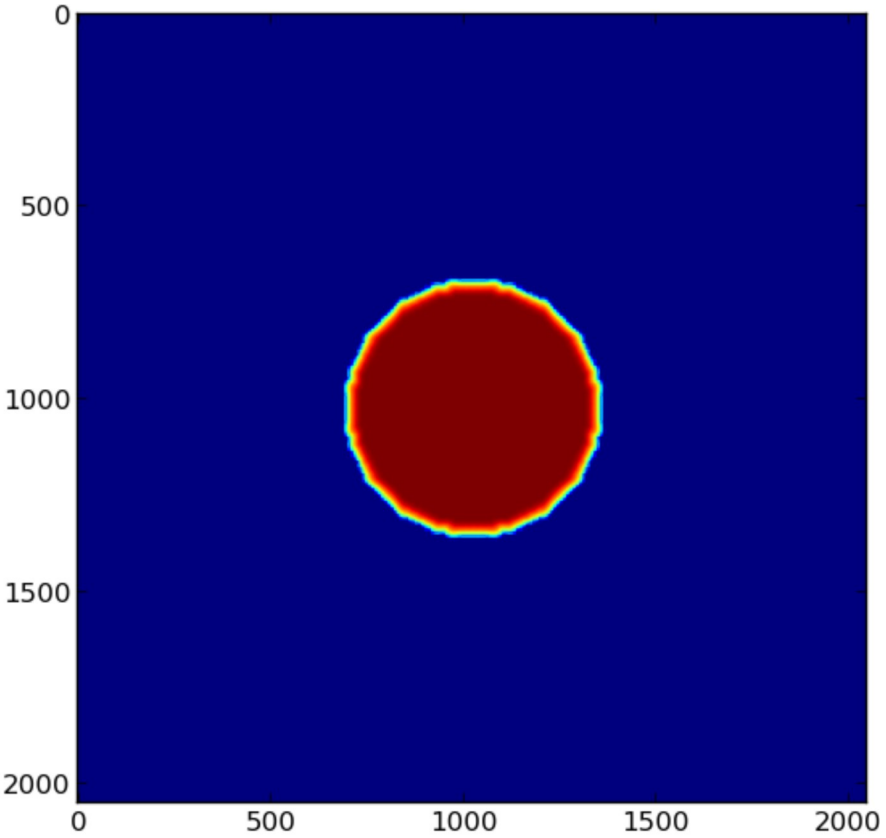


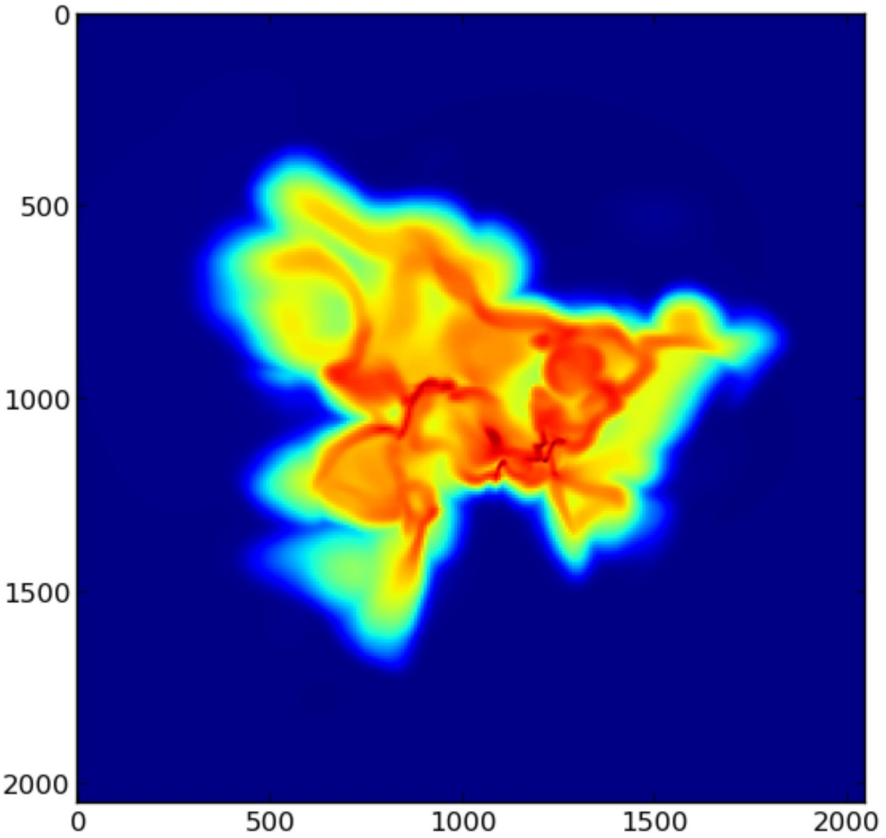
TO GENERATE A MOLECULAR CLOUD:

Uniform sphere of gas;

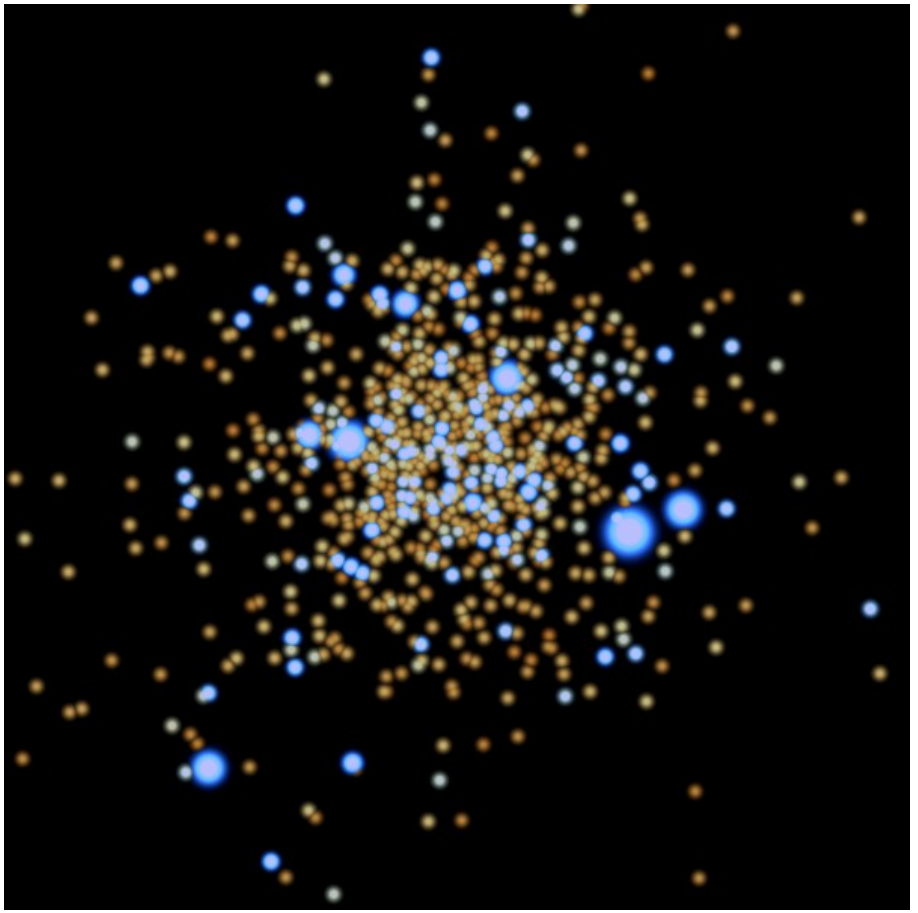
Total energy = 0 + Gaussian random motions;

Optional: seeded with turbulence as power spectrum.





TO GENERATE A STAR CLUSTER:



- Plummer or King profile for positions and velocities (already in starlab);

- Salpeter, Kroupa or similar IMF for masses;

- Fixed stellar radius or according to stellar evolution (SeBa, SSE);

- If stellar evolution, luminosities, temperatures and metallicity;

- Binaries: distribution for secondary masses, semi-major axis, eccentricities,...

TO GENERATE A GALAXY:

- Hernquist bulge as described;

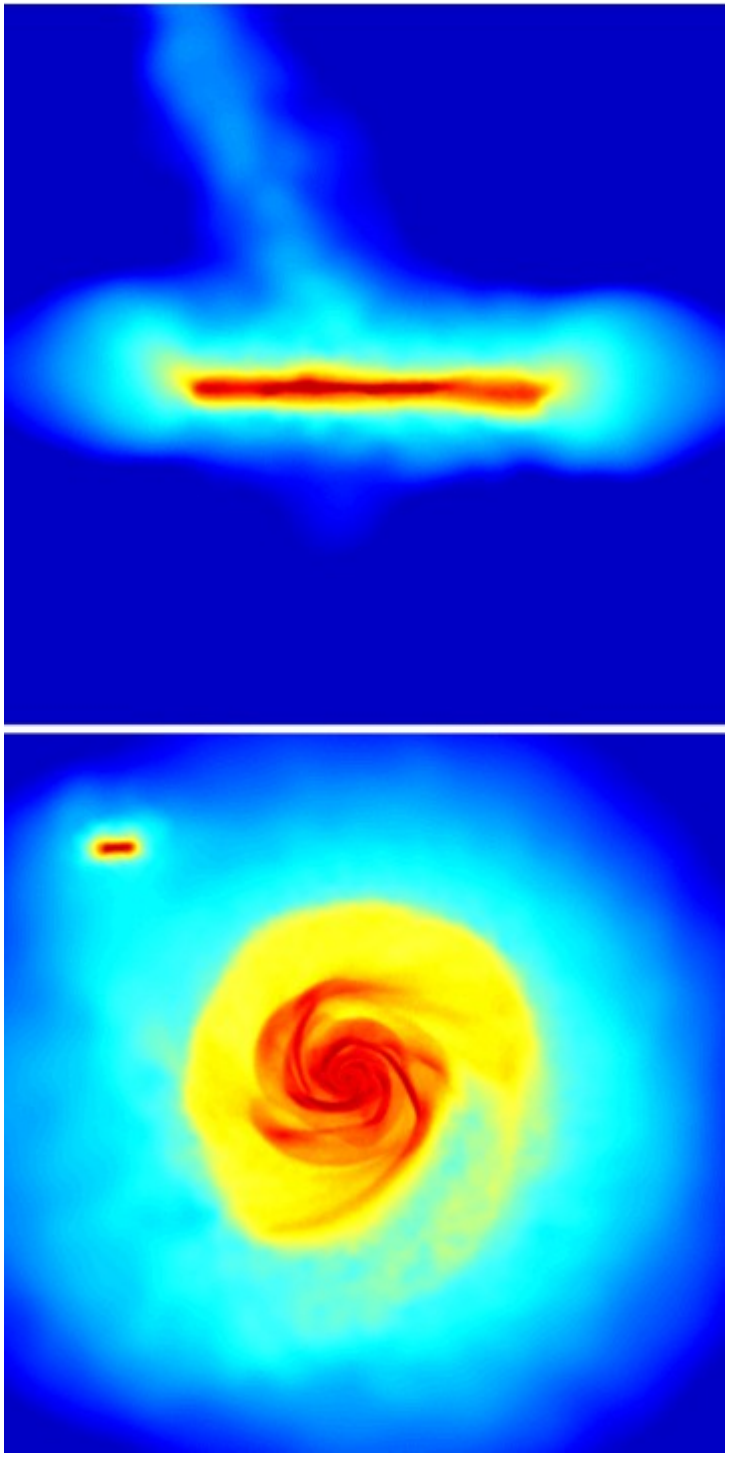
- Navarro, Frenk & White halo for dark matter;

- Eventually disc profile;

See e.g. Hernquist (1993),

Widrow & Dubinsky (2005),

ask me..



MM, Moore & Bland-Hawthorne 2008

COSMOLOGICAL SIMULATIONS:

Generate a cubic box with Zeldovich approximation

**SUGGESTION: use freely available grafic2 code,
developed by Edmund Bertschinger at MIT
see http://web.mit.edu/edbert**

Download grafic-2 tar file, read the README.txt (infos about settings), compile with Makefile

eg initial conditions are already in the right format for RAMSES