Random Numbers

Alex Rodriguez

arodrigu@ictp.it

~~Room: 265 2nd Floor~~

/afs/ictp/public/a/arodrigu/rand_num_2022.pdf

Why random numbers?

- direct Monte Carlo: physical process which contains randomness

  e.g.: radioactive decay, thermal motion

- (simple) Monte Carlo integration: multi-dimensional integrals

- Markov Chain Monte Carlo: simulation in Classical Statistical Mechanics,

  Quantum Mechanics etc.

True random numbers

- **Hard to obtain:**

  -) decay of radioactive isotopes

  -) atmospheric noise

  -) commercial products: e.g. QUANTIS → uses QM: photon transm. or reflected

  4 Mbits/s from a usb, too slow

- **Bad for computer simulations:** slow to generate, no reproducibility

  Applications: high quality/ low quantity

  requirements, e.g. cryptography

“Pseudo” random numbers

- Long sequences of numbers (x_1, x_2, x_3 …) generated by an algorithm on a computer
- Not truly random, completely deterministic

  ➔ if I know: $x_1, x_2, \ldots, x_n$ I can anticipate: x_{n+1}
- Reproducible: always the same sequence from the same seed x_1

- Look “nearly random”: have statistical properties similar to true r.n.

e.g.: uniform distribution P(x) = 1 if 0 <= x < 1, =0 otherwise

<u>Correlations:</u>

$$P(x_n, x_{n+j}) \cong P(x_n) P(x_{n+j})$$

$$P(x_n, x_{n+j}, x_{n+k}) \cong P(x_n) P(x_{n+j}) P(x_{n+k}) \text{ and so on and so forth}$$

<u>Ideally (no correlations) we have equalities!</u>

Generators of uniform "pseudo" random numbers

Modulo operation:
a mod *b* := reminder of integer division a/b

Linear congruent generators (Lehmer,1951)

Produce a sequence of integer numbers using the function:

$$Y_{n+1}=(aY_n + c) \bmod m$$

- The first element Y_1 **(called "the seed")** must be initialized
- Maximum is m-1
- Finite period, then sequence repeats itself.
- At most *m-1* different numbers (0 is ruled out) obtained with special combinations of *a,m*
- $X_n = Y_n$/m is in the interval [0:1) (!NOTE: REAL division)

Period (and quality) depends sensitively on ***a, c, m***.

- worst choice: if c = 1, a = 12, m = 143 period is 2
- a good choice (a and m are coprime): $c = 0$, $m = 2^{31}-1 = 2147483647$, $a = 16807$

  ➔ *max. period is* $2^{31}-2 \approx 2x10^9$

•LCG is simple, fast but period is not long and correlation (especially in triples)

Nowadays (much) better algorithm exist: use them for scientific/industrial applications

Fortran Intrinsic Random Number generator

```
CALL RANDOM_NUMBER(r)
```

- ‘`r`’ must be a real valued variable (single o double precission).
- ‘`r`’ can be an array (discoraged).
- For scientific simulations you must learn how to fix the seed in order to get reproducibility (first element of the series).

Fortran Intrinsic Random Number generator. Setting the seed.

```
integer :: n
integer,allocatable :: seed(:)
real :: r
call random_seed(size=n)
allocate(seed(n))
seed = 123456789    ! putting
arbitrary seed to all elements
call random_seed(put=seed)
deallocate(seed)
CALL RANDOM_NUMBER(r)
```

- We learned how to generate random numbers between [0,1)

- Can get other uniform distributions,

  uniform x in [a,b):

  $x=a+(b-a)u$ where u is a r.n. in [0,1)

Non-uniform random numbers

Q: How can we get a random number x distributed with a probability distribution f(x) in the interval $[x_{min},x_{max}]$ from a uniform random number u?

Transformation method

- Create random numbers with distribution ***f(x)***

  f(x)dx: probability of producing r.n. between x and x+dx

- We need a relation in the form $x = G(u)$

  u is a uniform random number [0:1)

Take the inverse,

$$u = G^{-1}(x)$$

and differentiate,

$$du = [G^{-1}(x)]' \, dx = f(x) \, dx$$

Note: inverse

=> $G^{-1}(G(x)) = x$

Transformation method

Integrate, $u = G^{-1}(x) = \int_{x_{\min}}^{x} f(x')dx'$

- Note that $G^{-1}$is the definition of the *cumulative distribution function*.
- We obtain the functional relation between u and x.
- Still need to invert the function to get G(u).

Transformation method

Example: Exponential distribution:

$$f(x) = e^{-x} \longrightarrow u = \int_0^x f(x')dx' = 1 - e^{-x}$$

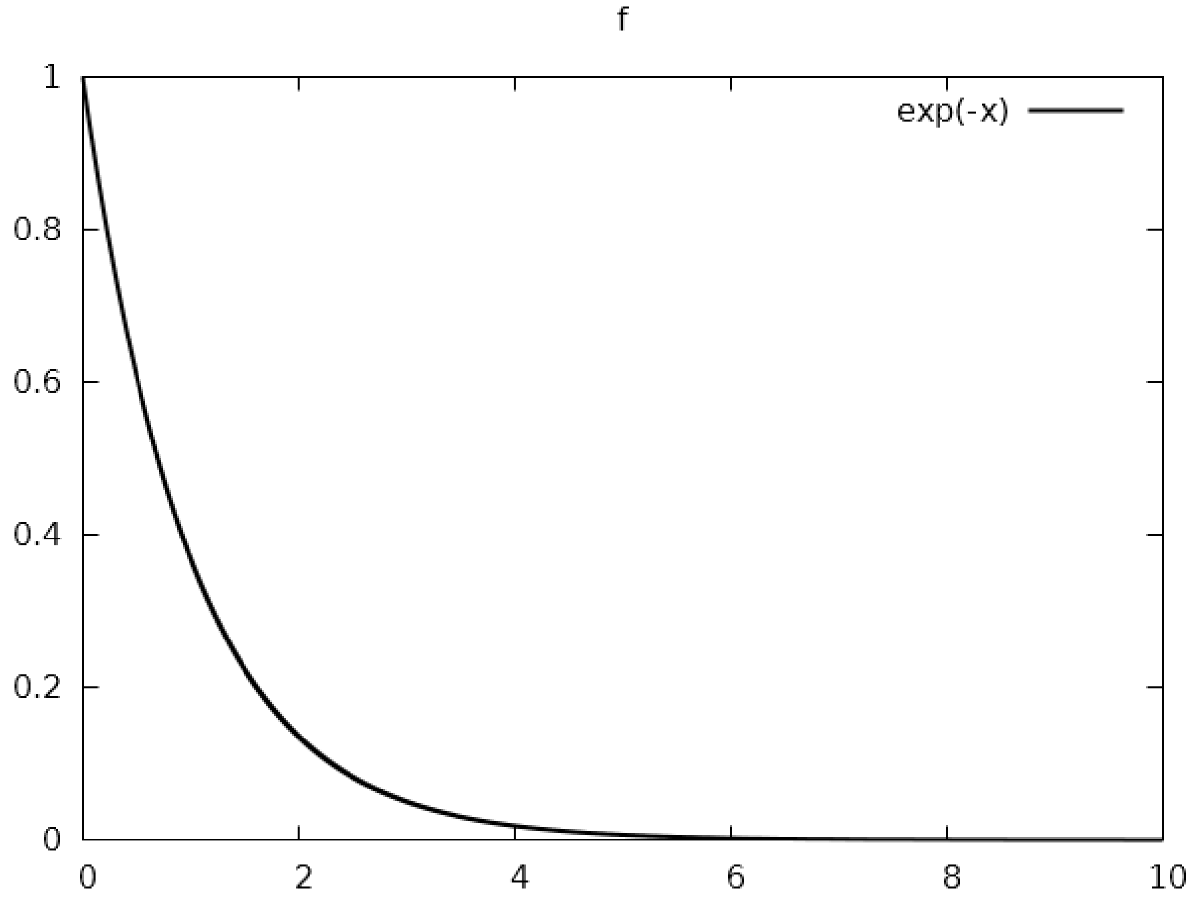
Inverting ➔ $x = -\log(1-u)$

Sample uniform r.n in [0:1): ***u*** (e.g., using linear congruent generator or better…)

Calculate: ***x = -log(1-u)***

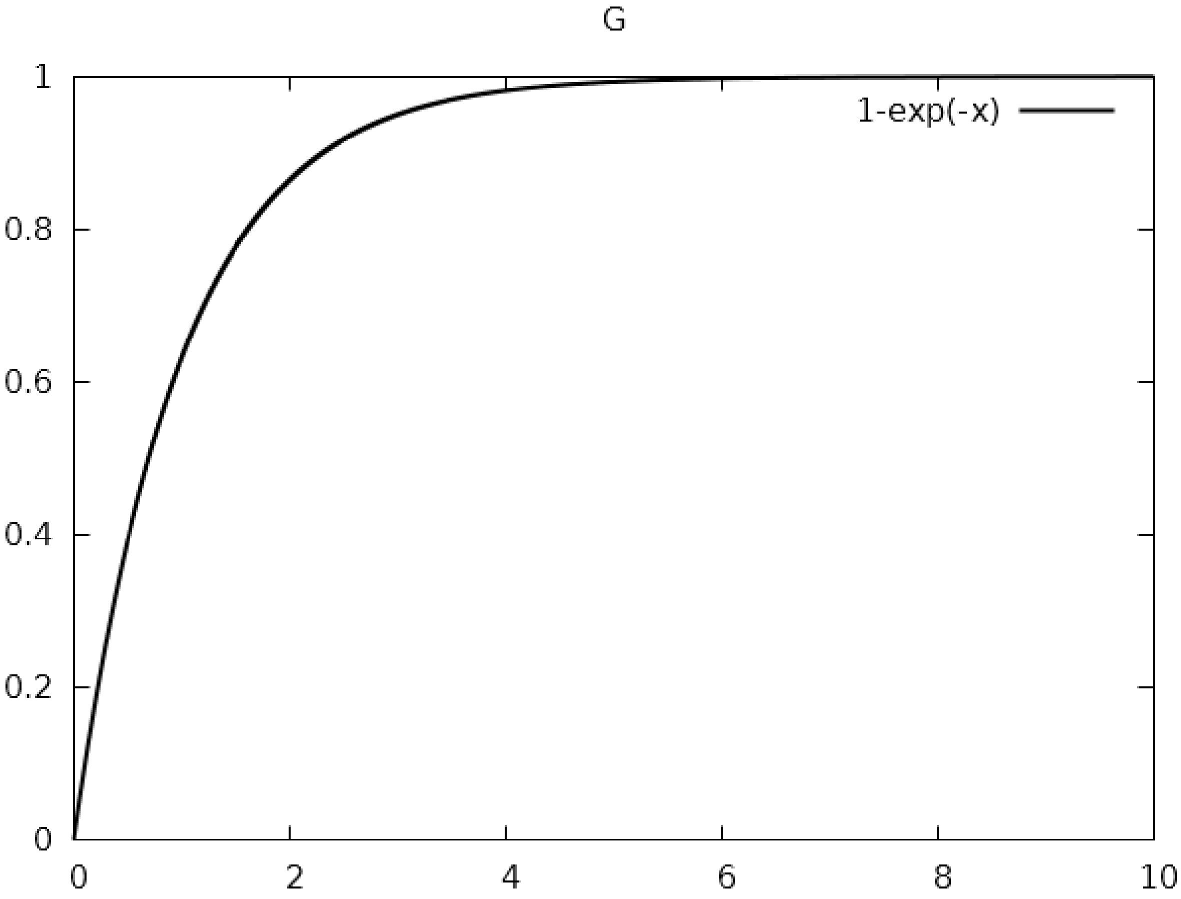
➔ ***x*** is in (0:oo) distributed according to ***e***$^{-x}$

Understanding the Transformation method.



Probability distribution function

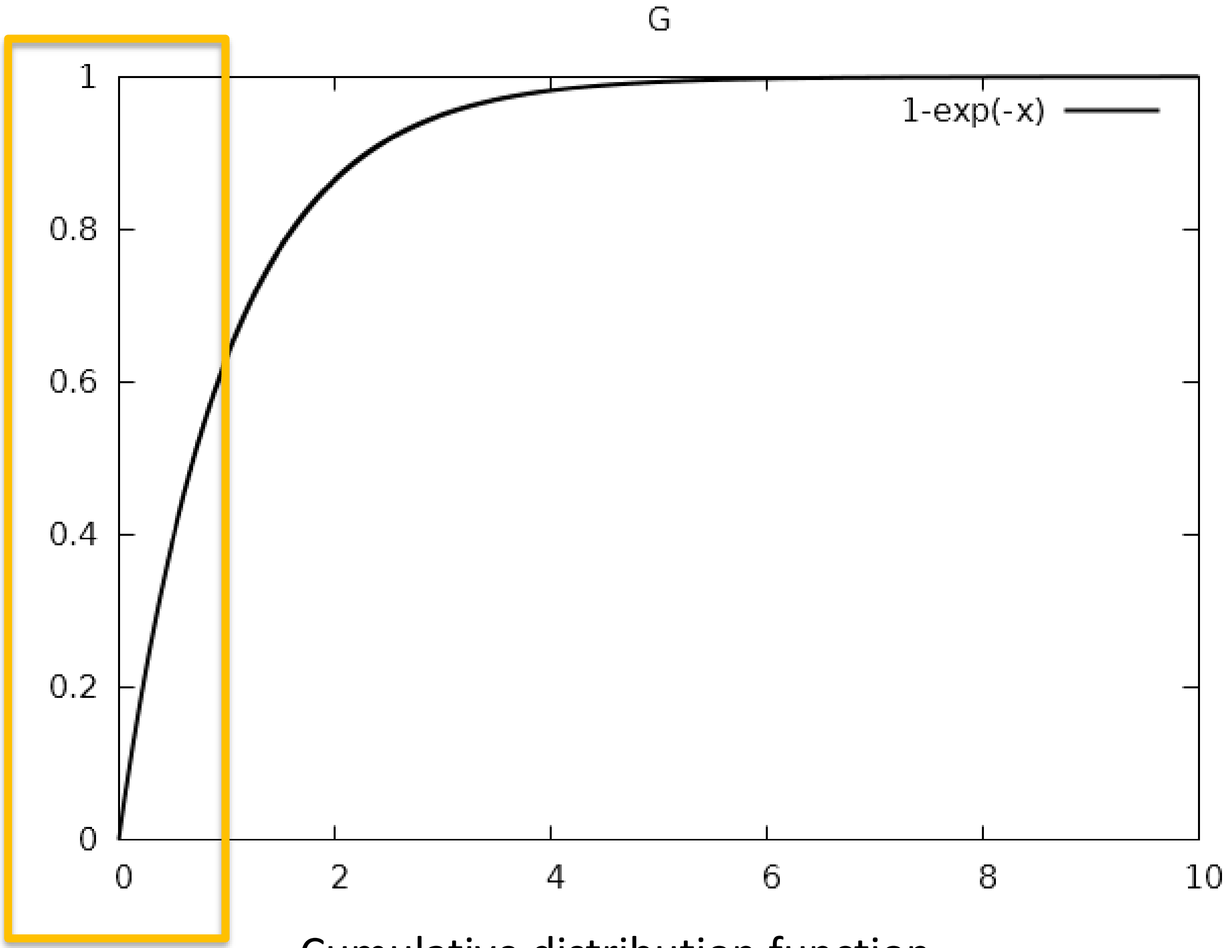
Understanding the Transformation method.



Cumulative distribution function

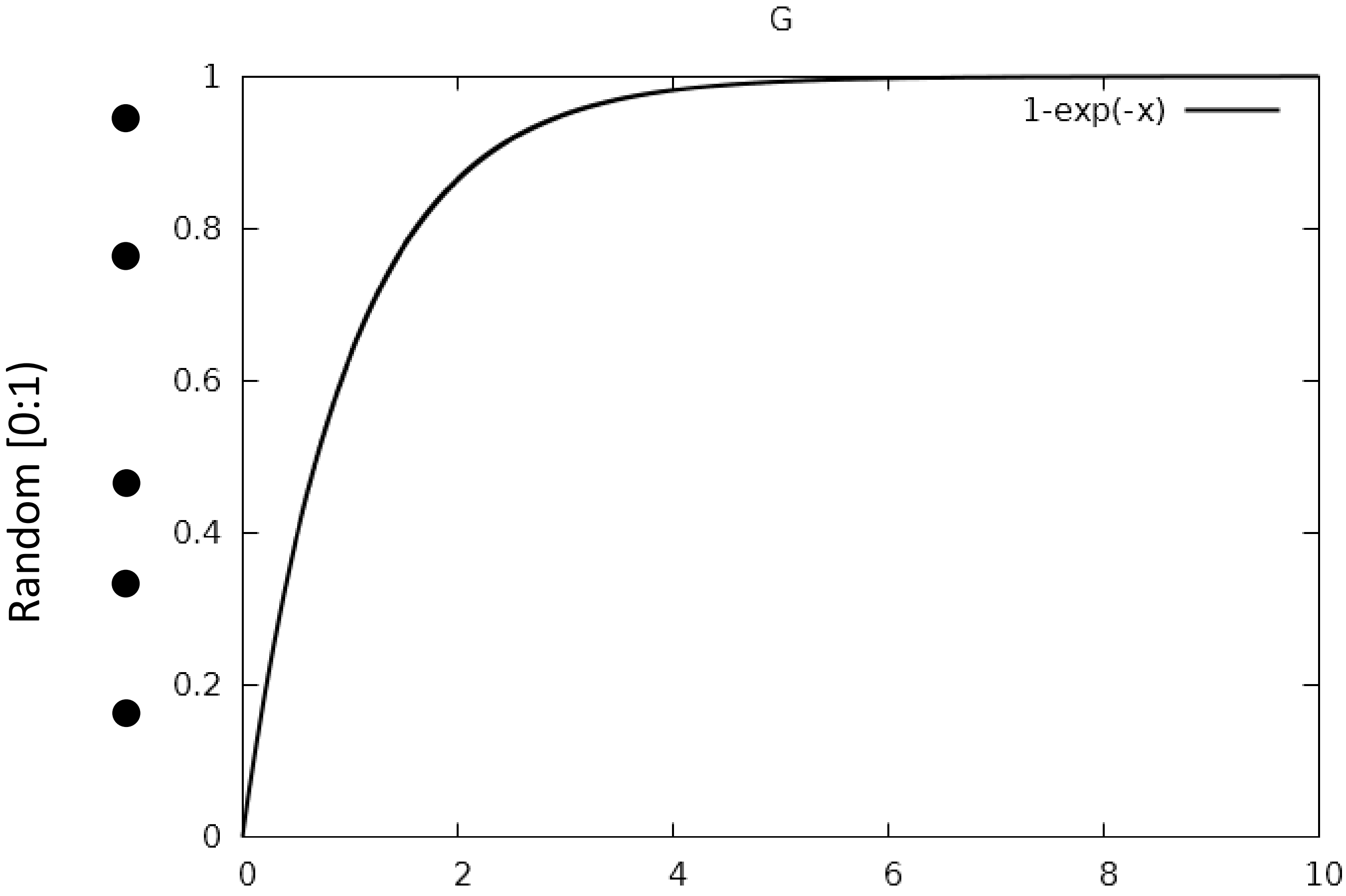
Understanding the Transformation method.

By the definition of the cumulative distribution function, this interval is [0,1)

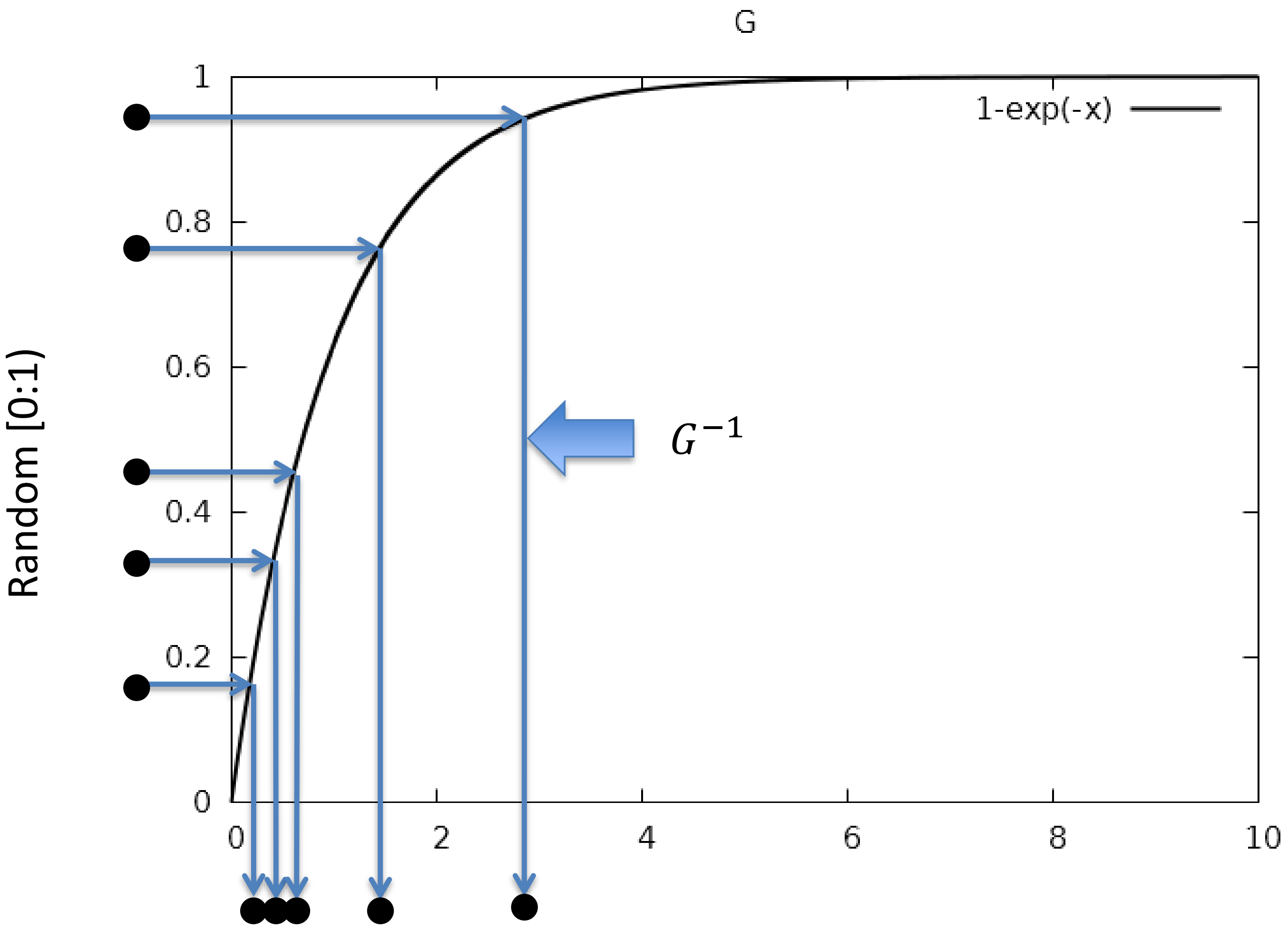


Cumulative distribution function

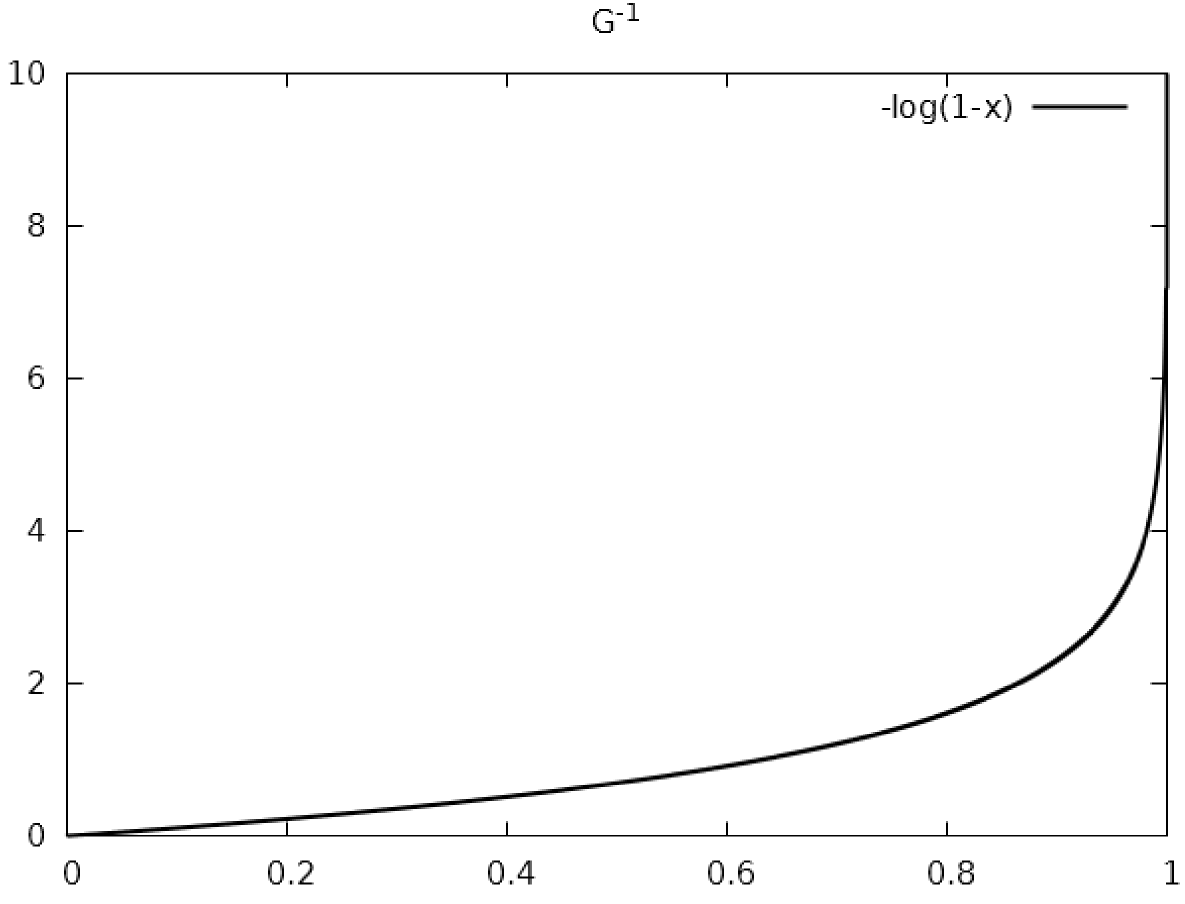
Generate random numbers in the domain of the CDF [0:1)



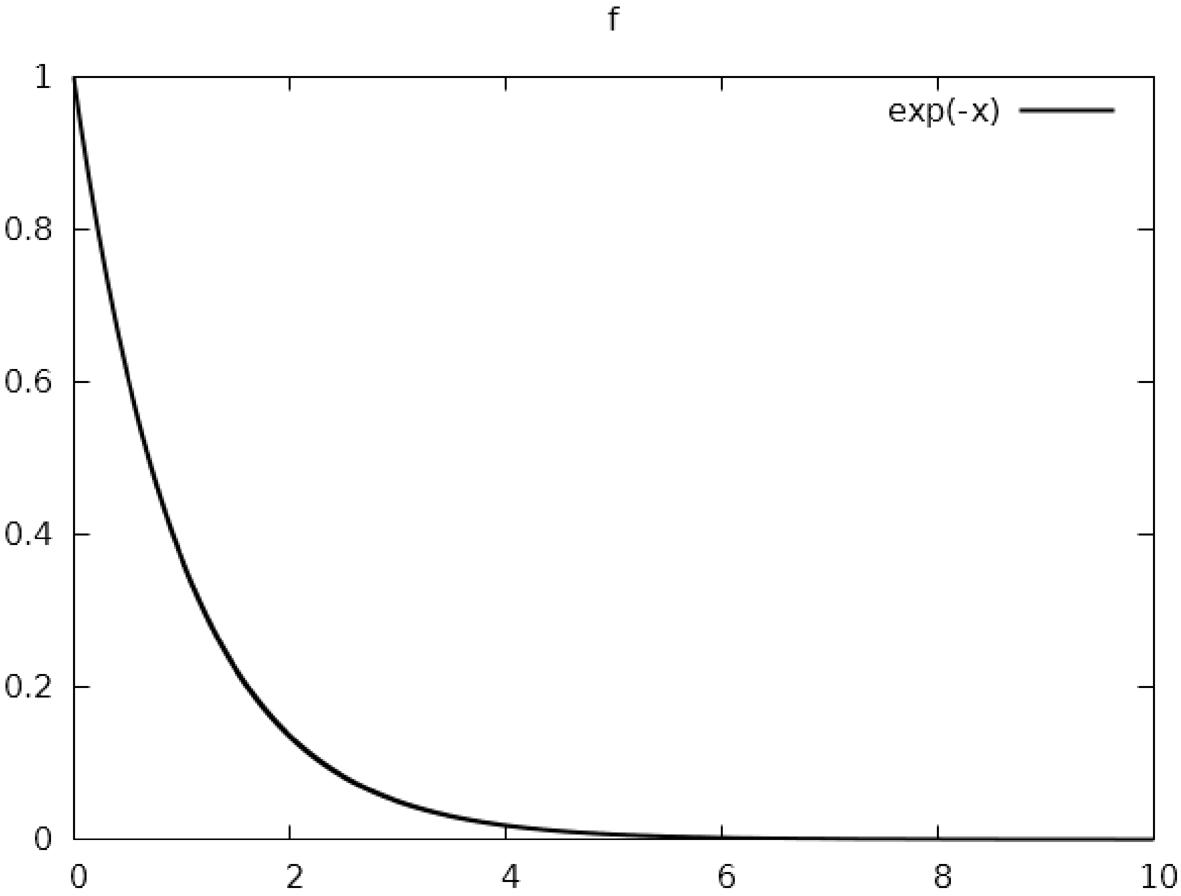
Apply the inverse tranform

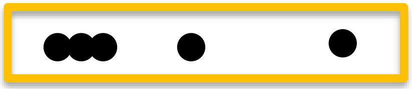


In this case, G^{-1} is analytical



Result:





These numbers follow the original distribution function

Transformation method

IMPORTANT EXAMPLE: Gaussian distribution (Box-Muller method)

Impossible to invert u=G^{-1}(x) in 1-d but possible in 2-d

$$f(x,y)=\frac{1}{2ps^{2}}\exp\text{æ ç è}-\frac{x^{2}+y^{2}}{2s^{2}}\text{ö ÷ ø}$$

2D Gaussian distribution:

$$f(x,y)dxdy=\frac{1}{2\pi\sigma^{2}}\exp\left(-\frac{R^{2}}{2\sigma^{2}}\right)RdRd\theta$$

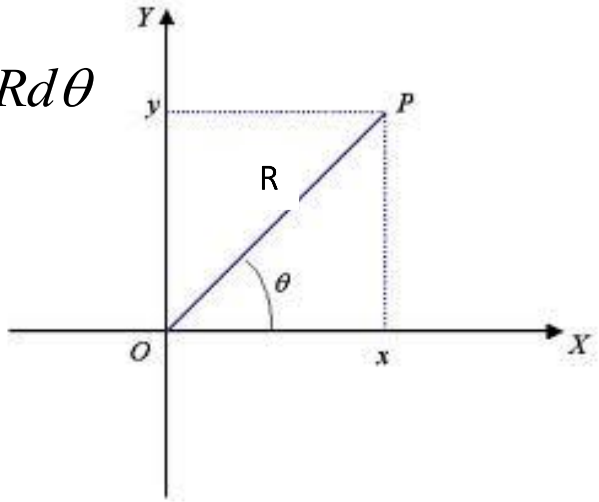
in polar coordinates:

$$u=-\exp\left(-\frac{R^{2}}{2\sigma^{2}}\right)+1$$

integrating from 0 to R:

$$\boxed{R=\sqrt{-2\sigma^{2}\log(1-u)}}$$

inverting, we find:



- Extract r.n. from uniform distribution: u
- Calculate R
- Extract r.n. from uniform distribution: u’
- Calculate θ = 2πu’

➢ x and y are Gaussian random variables

$$R=\sqrt{-2\sigma^{2}\log(1-u)}$$

$$\theta=2\pi u'$$

$$x=R\sin\theta \qquad y=R\cos\theta$$

Rejection method

- If the inverse of the integral cannot be found
- Less efficient (because of rejections)
- works only if the function has a finite bound

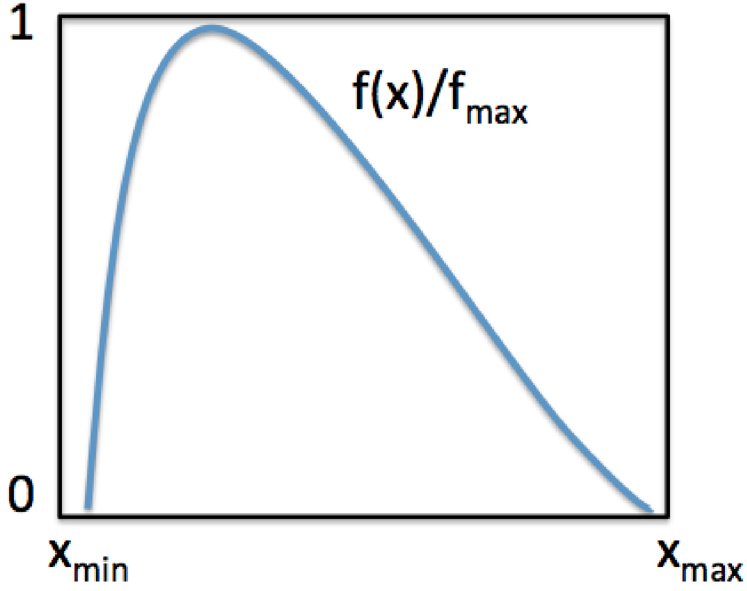
1) generate a random number x uniformly distributed in $[x_{min},x_{max})$.

2) generate another r.n., call it r, in [0,1)

3) If: $r < f(x)/f_{max}$ accept x

otherwise: reject and go to step 1

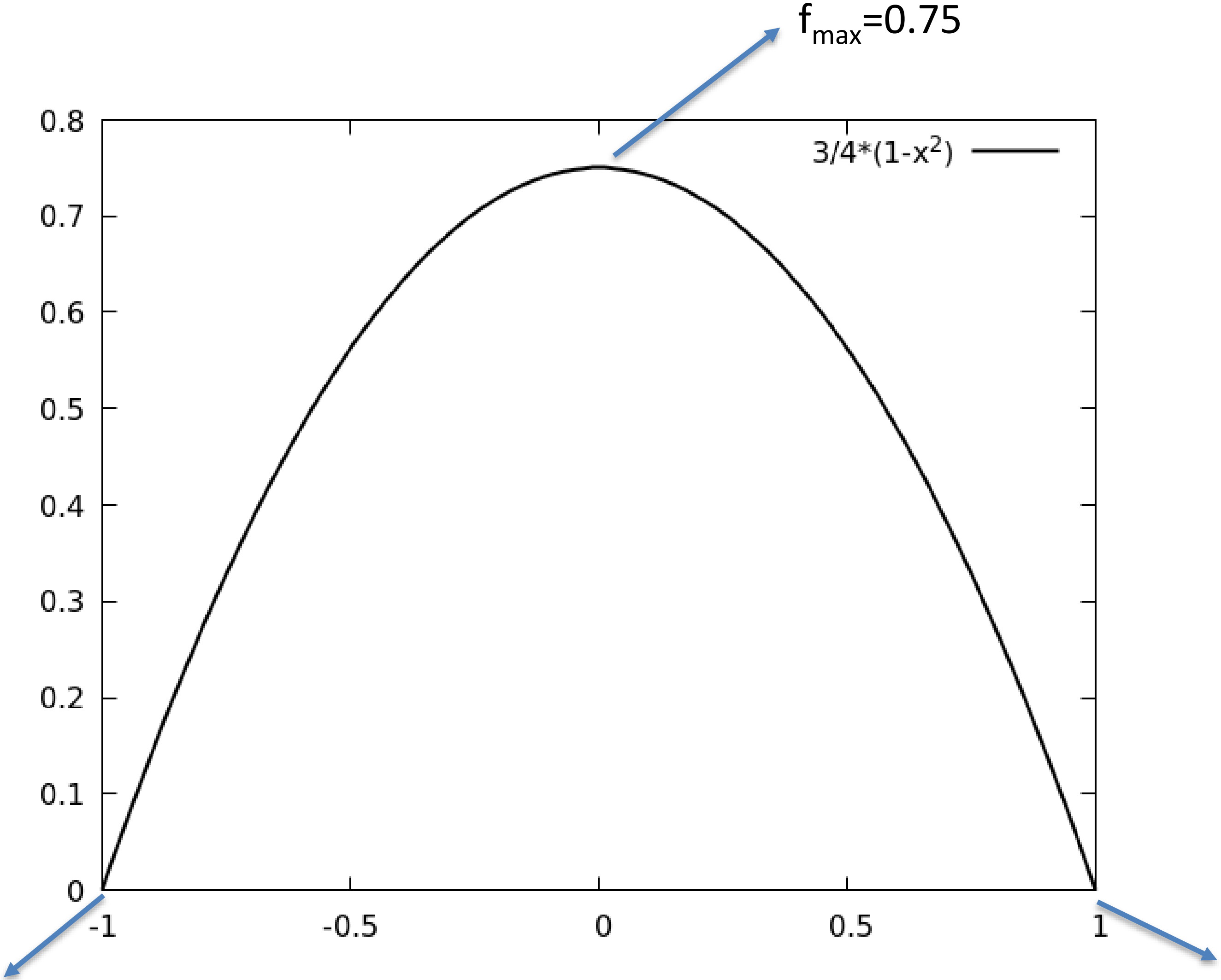
$f_{max} = \max(f(x))$



Understanding the Rejection method.

- It works because the probability of acceptance of the generated random numbers is proportional to the Probability Density Function.
- We need the interval and the maximum value of the Probability Density Function in this interval.

Understanding the Rejection method.



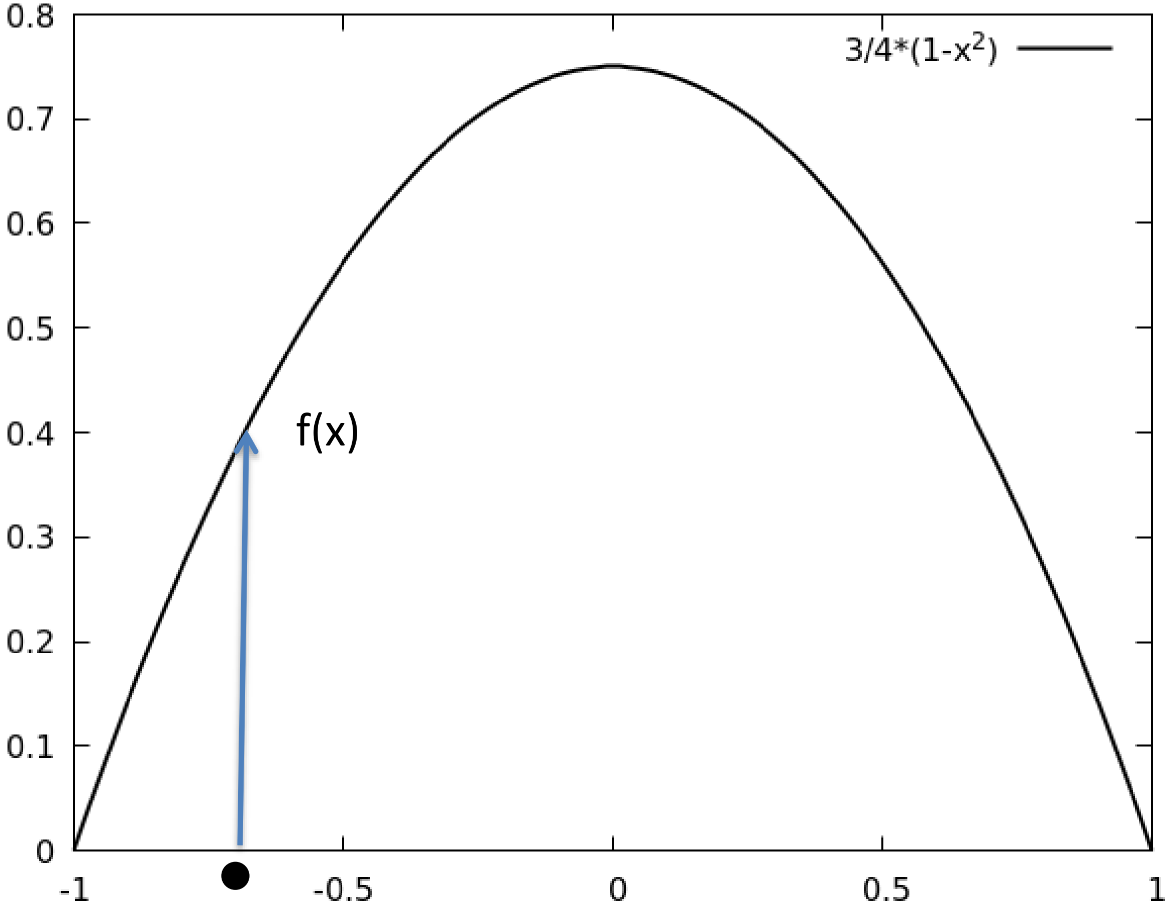
f_{max}=0.75

x_{min}=-1

Probability distribution function

x_{max}=1

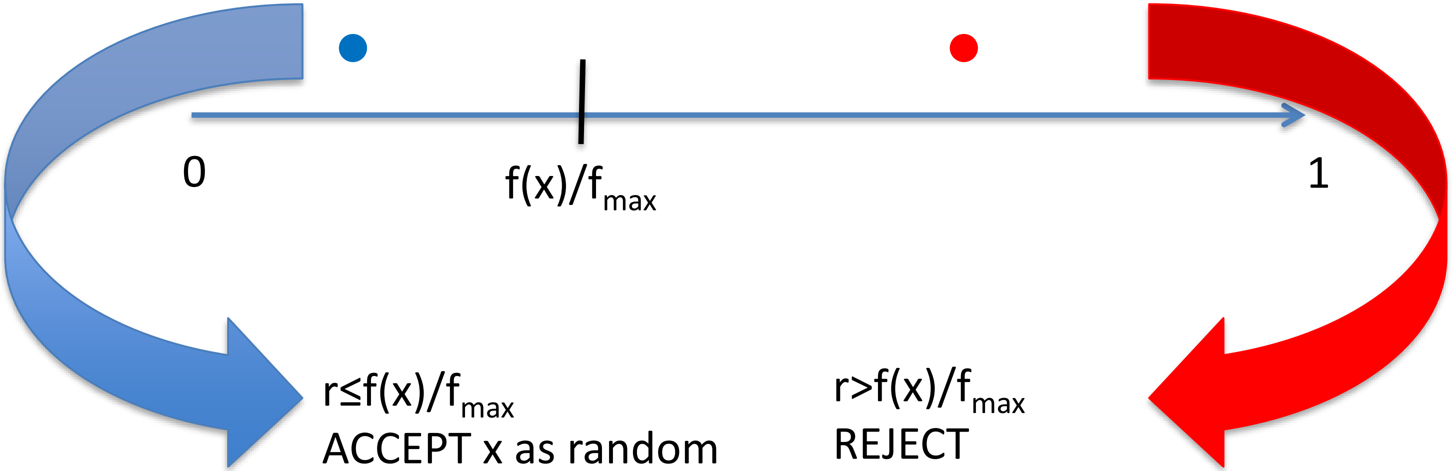
Random number uniformly distributed in the interval between x_{min} & x_{max}



Probability distribution function

New random number with acceptance proportional to f(x)

r (new random number) between 0 and 1.



0

$f(x)/f_{max}$

1

$r \leq f(x)/f_{max}$
ACCEPT x as random number with the pdf

$r > f(x)/f_{max}$
REJECT

NOTE:

- If f_{max} is bigger than the real maximum of the function in the interval, the method will still work but less efficiently (decrease the acceptance ratio).
- Rejected points do not count when generating random numbers, so if you need N random numbers you need N ACCEPTED points (use `DO WHILE`).

Assignment

- Make a program that generates 10000 points in the interval [0,1) from the distribution $f(x) = 3x^2$ with both the transformation and the rejection methods.
- Make a subroutine implementing the Box-Muller method.