1. Invent Yourself

Martin Marek

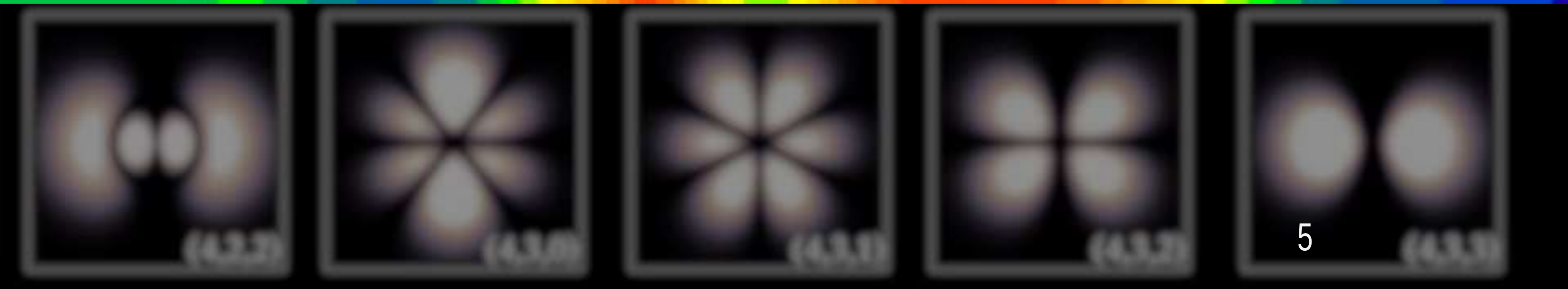
Problem definition

"Truly random numbers are a very valuable and rare resource. Design, produce, and test a mechanical device for producing random numbers. Analyse to what extent the randomness produced is safe against tampering."

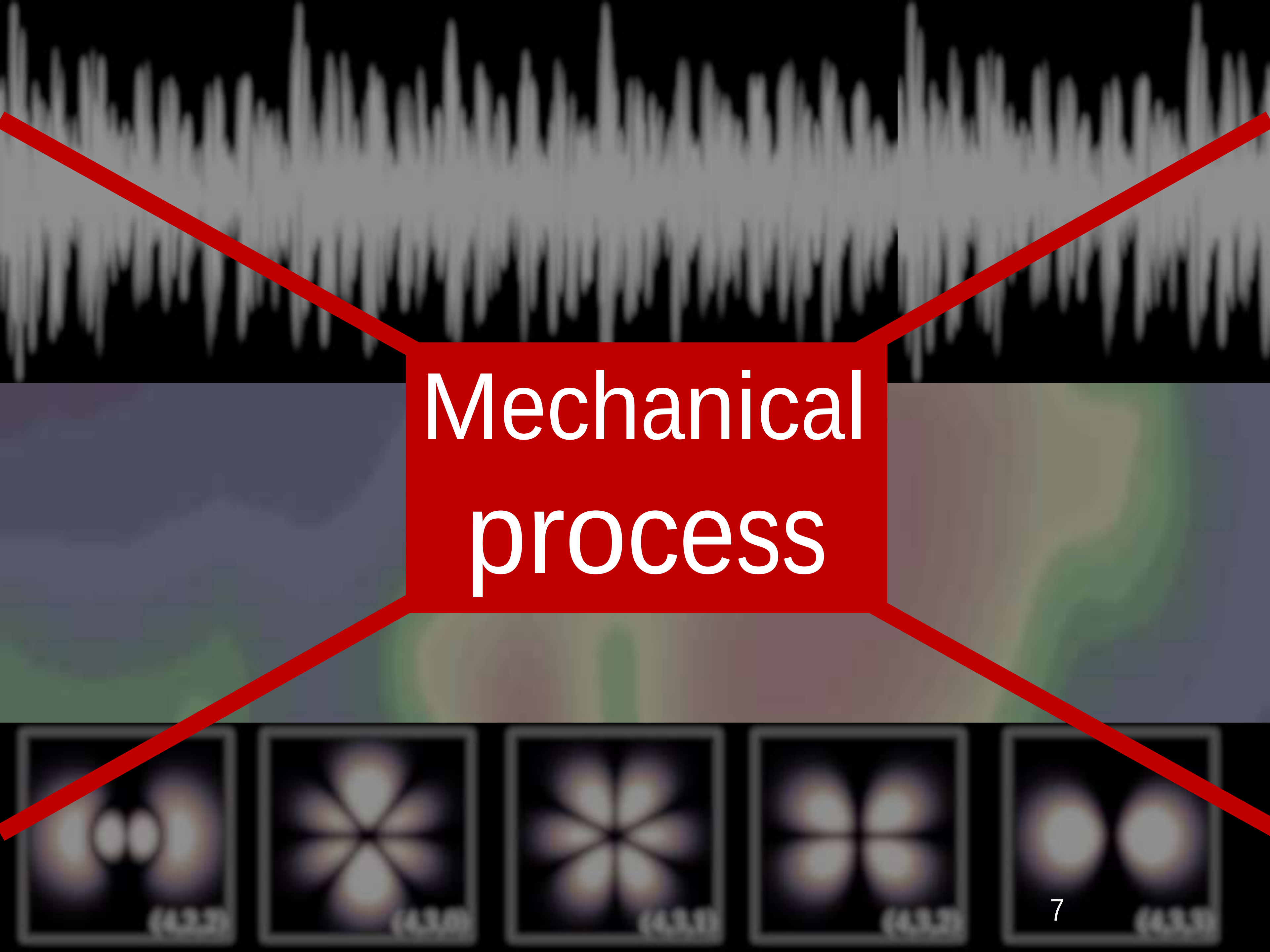
MECHANICAL DEVICE DEFINITION

Atmospheric noise

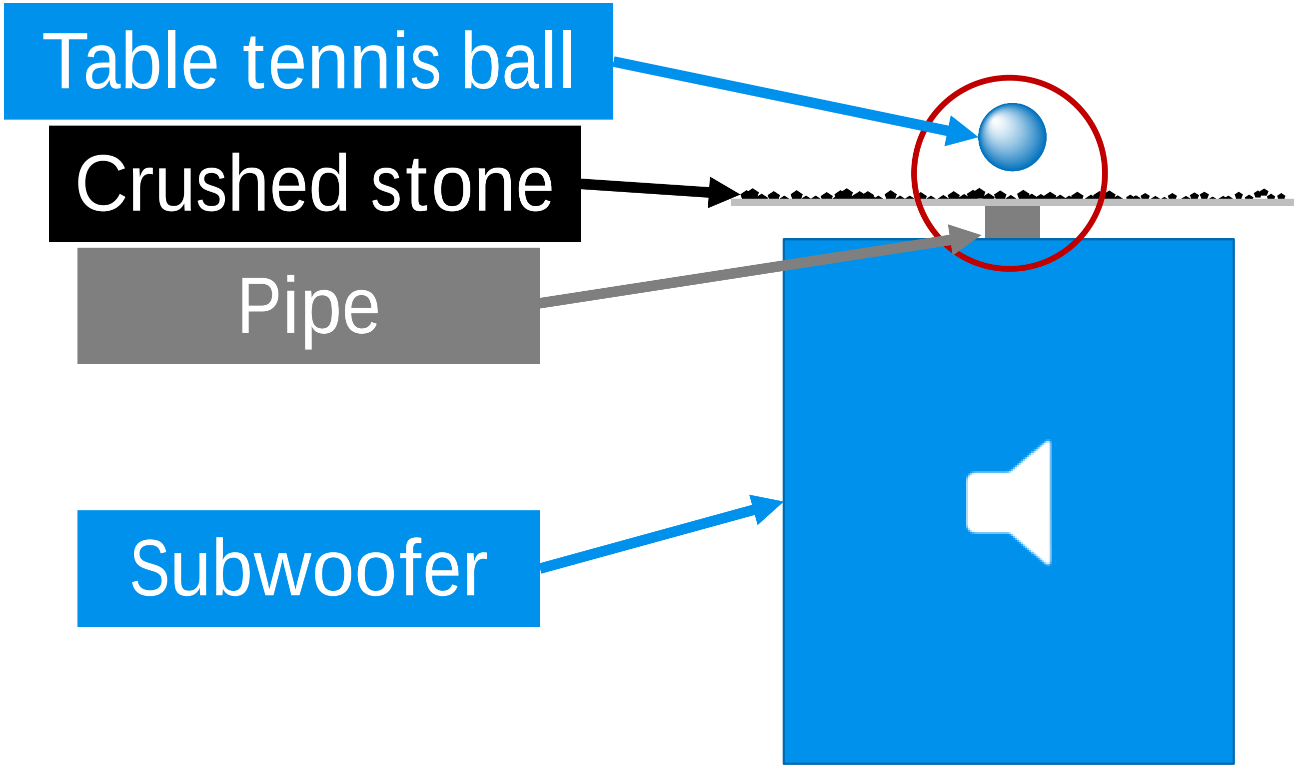
Thermal noise

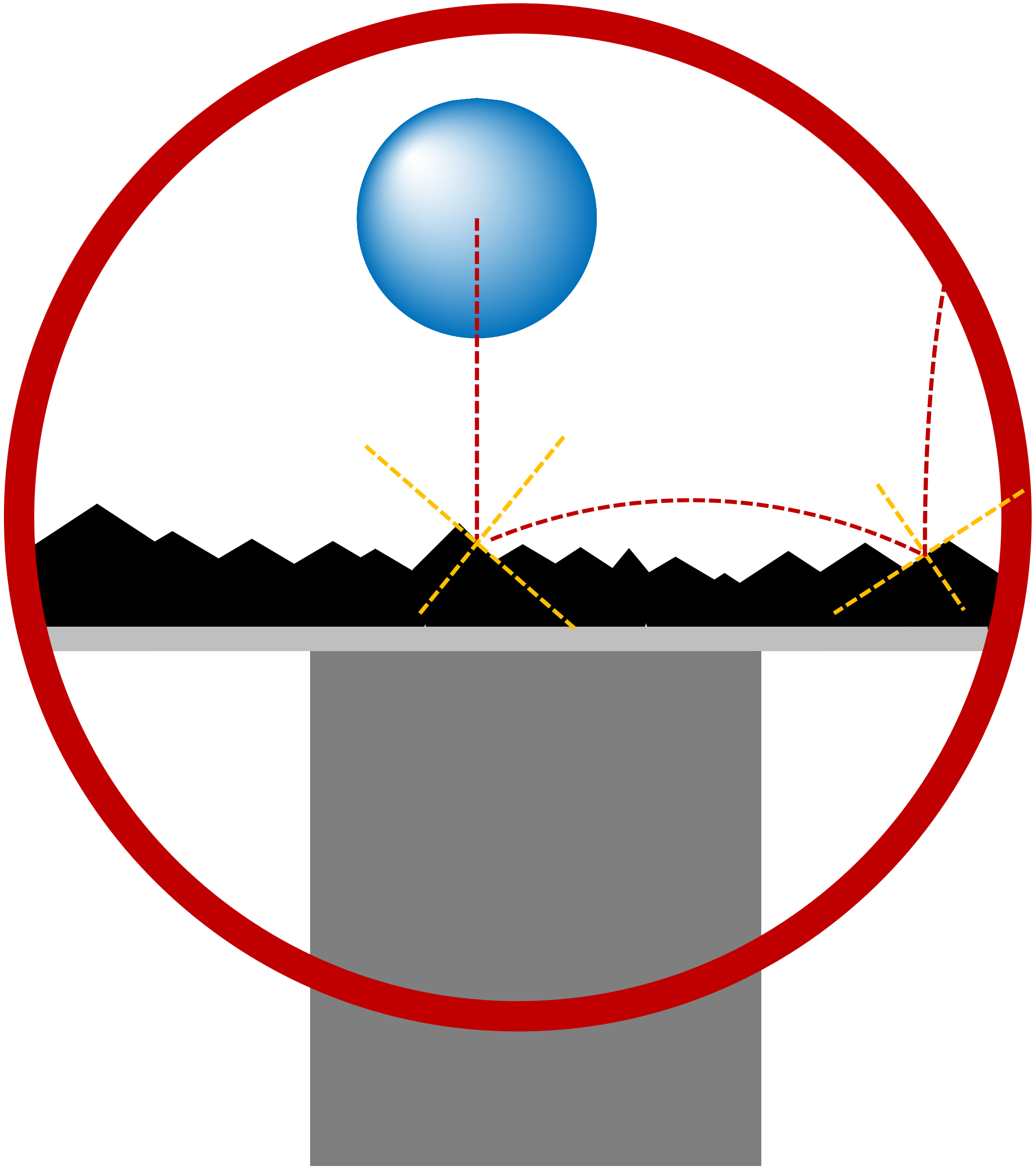


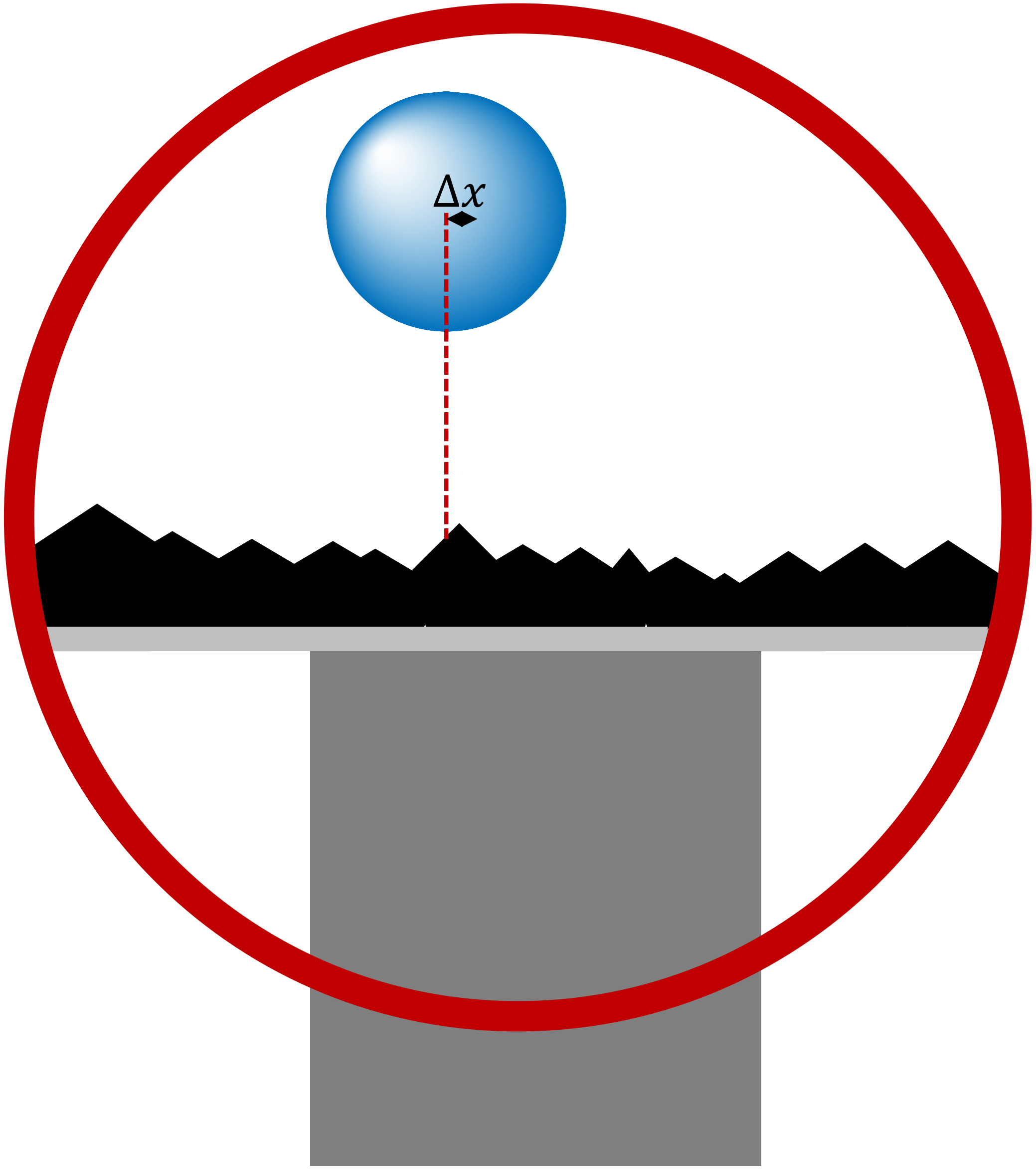
QM system

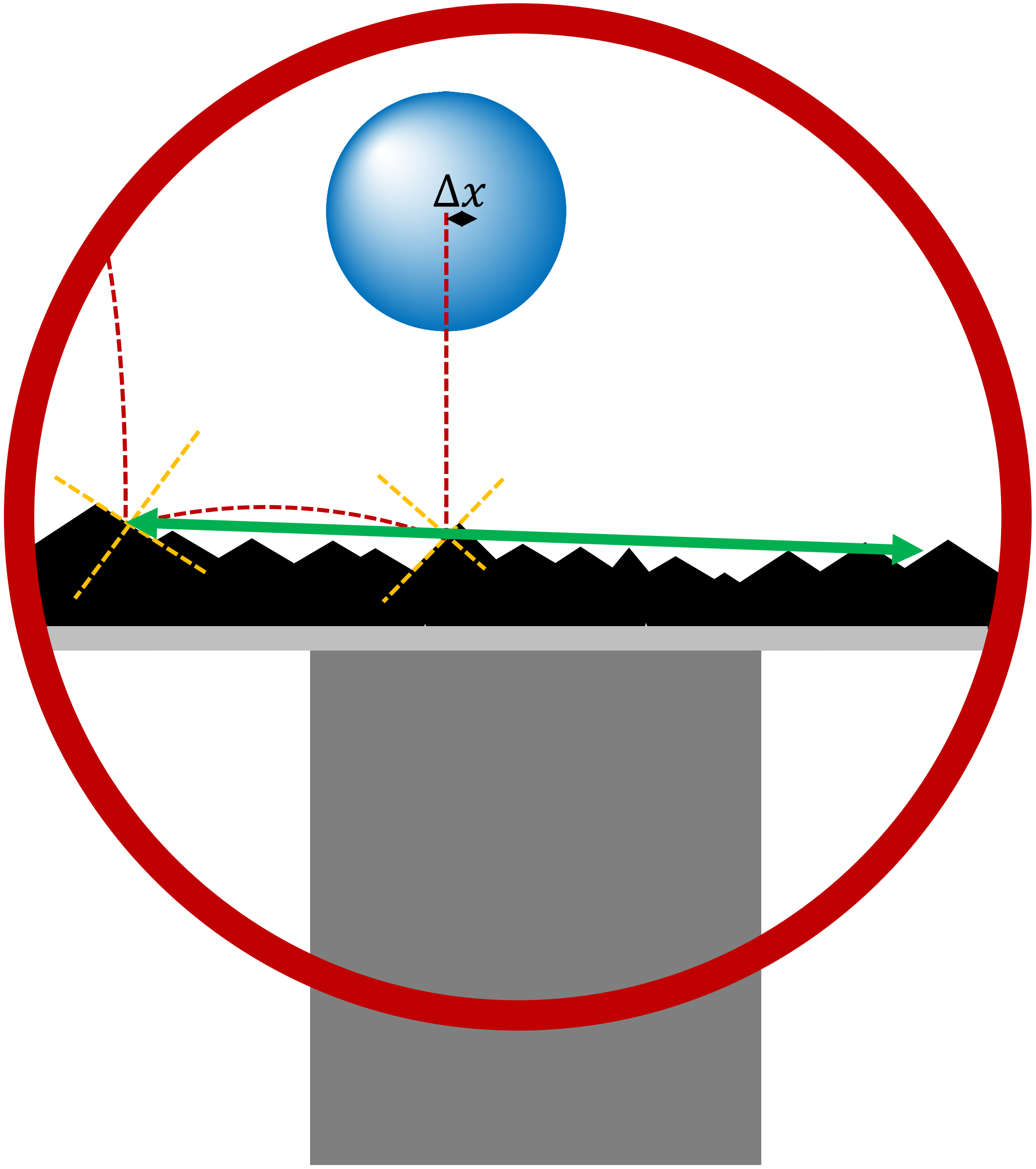

Mechanical
process

Chaotic Mechanical System



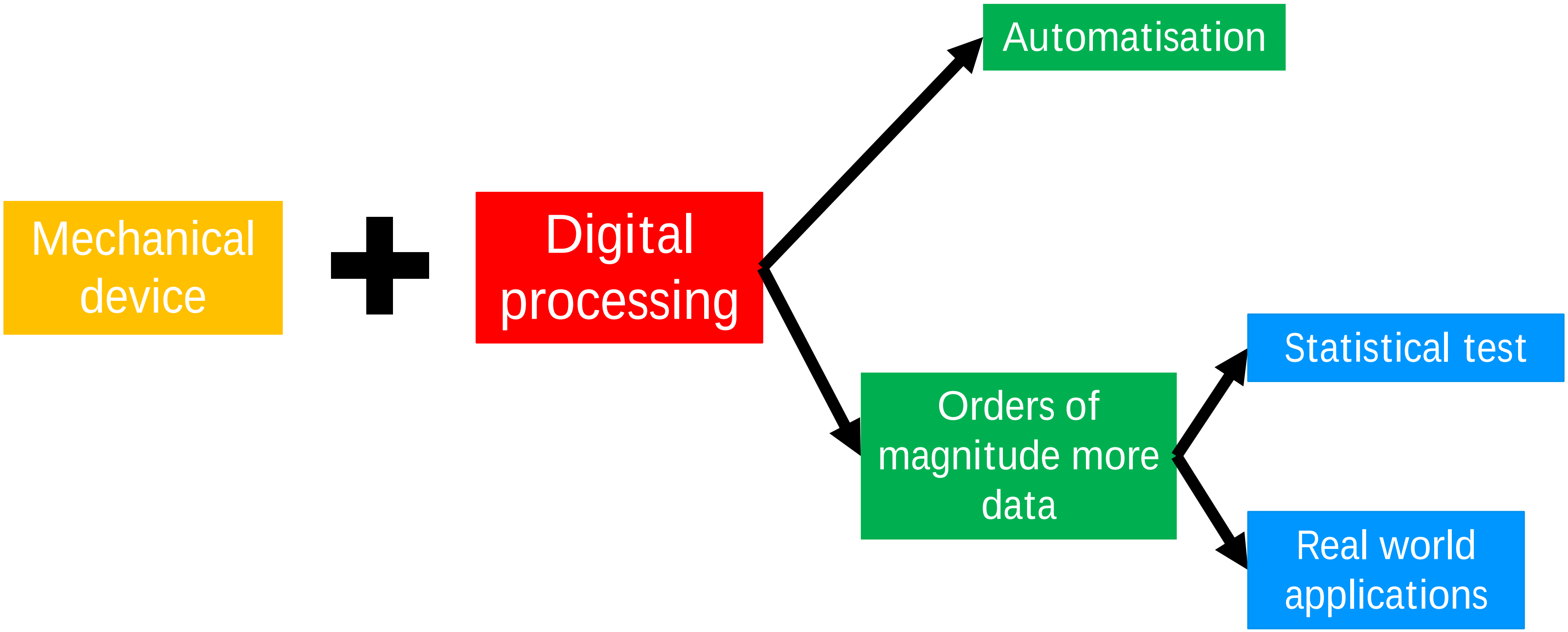


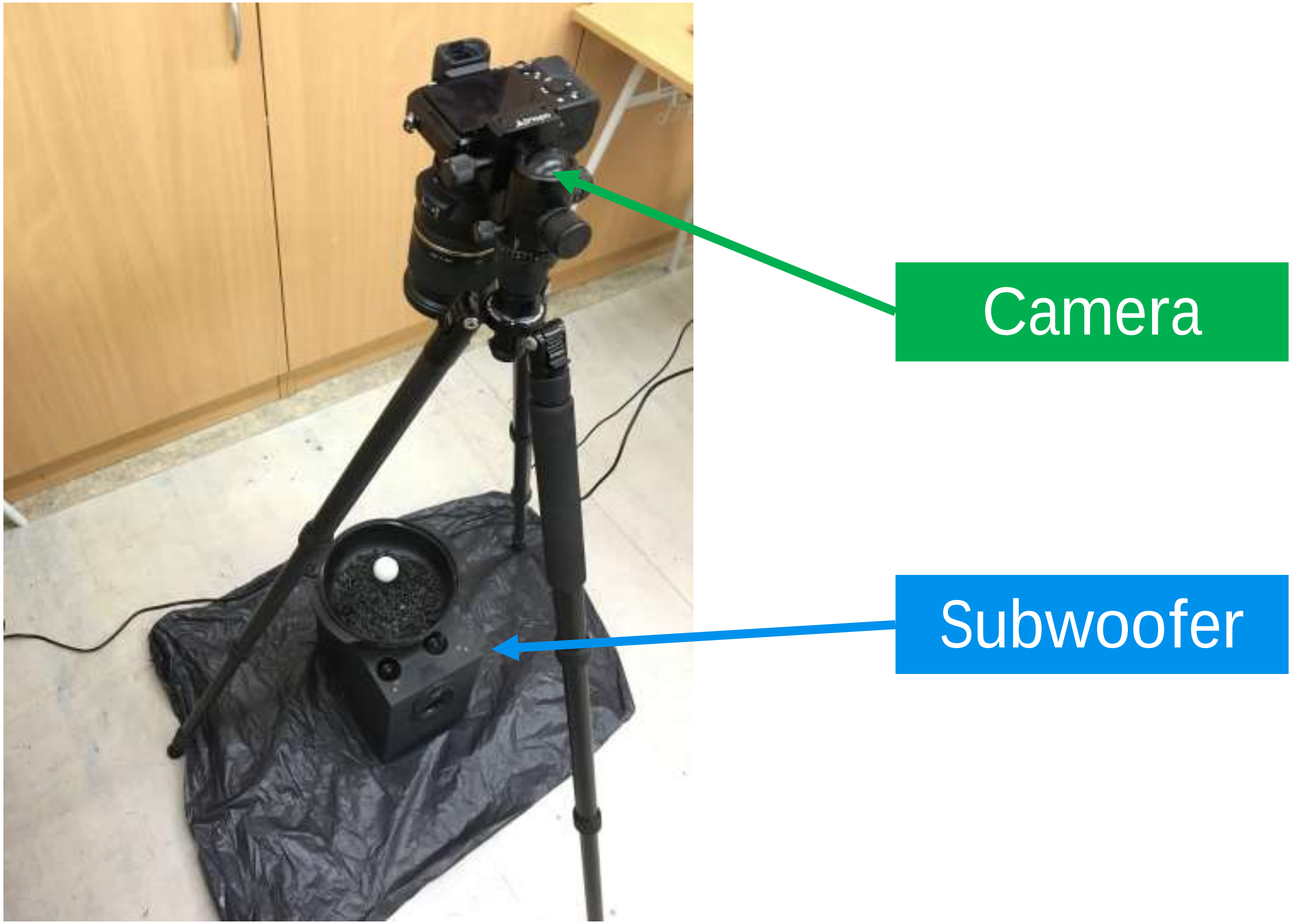

Δx



Chaotic system

Digital processing



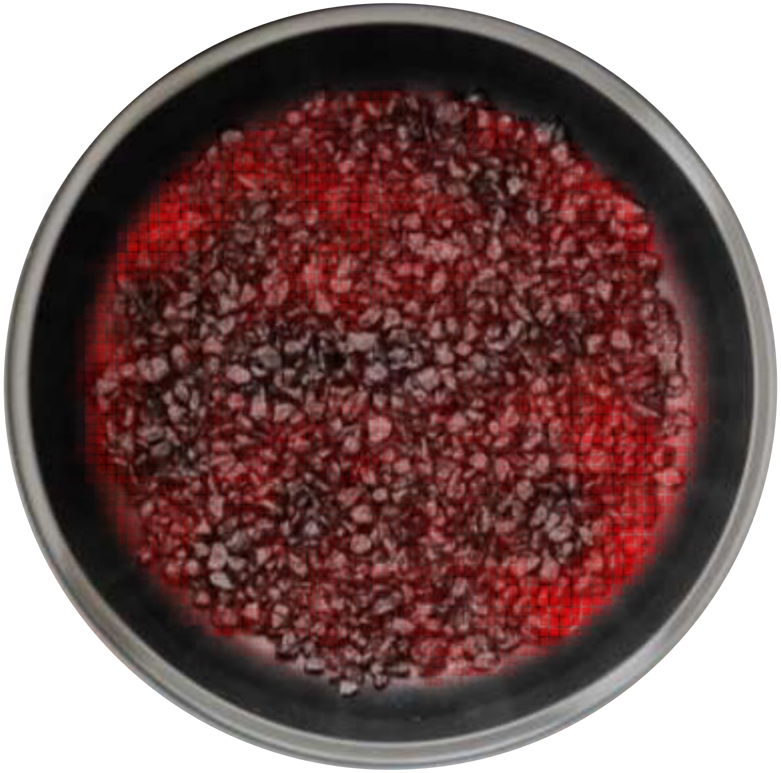

Camera
Subwoofer

Exemplary video

POSSIBLE WAYS OF GENERATING AN OUTPUT

Position

Position



~~x, y~~

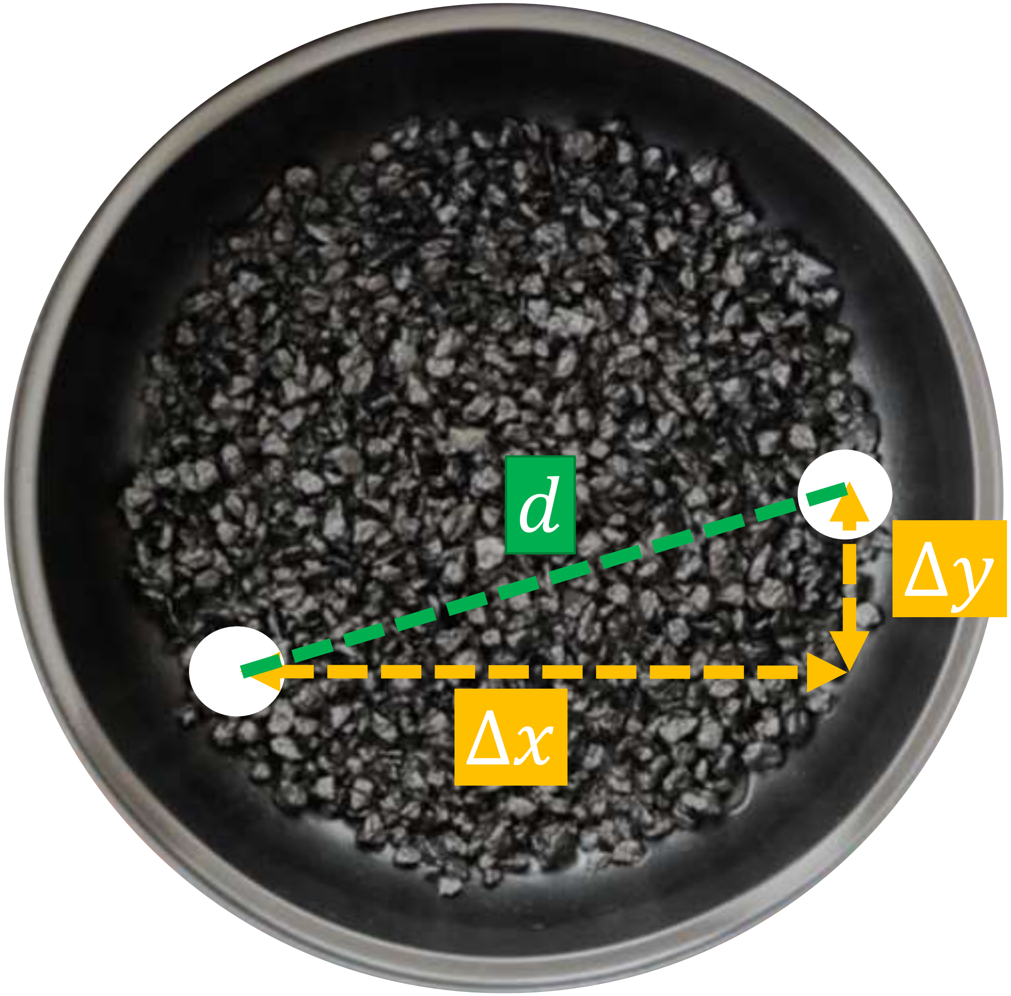
Distance

$$\Delta x \Delta y \leq 2r^2$$

Correlated

Data needs to be combined

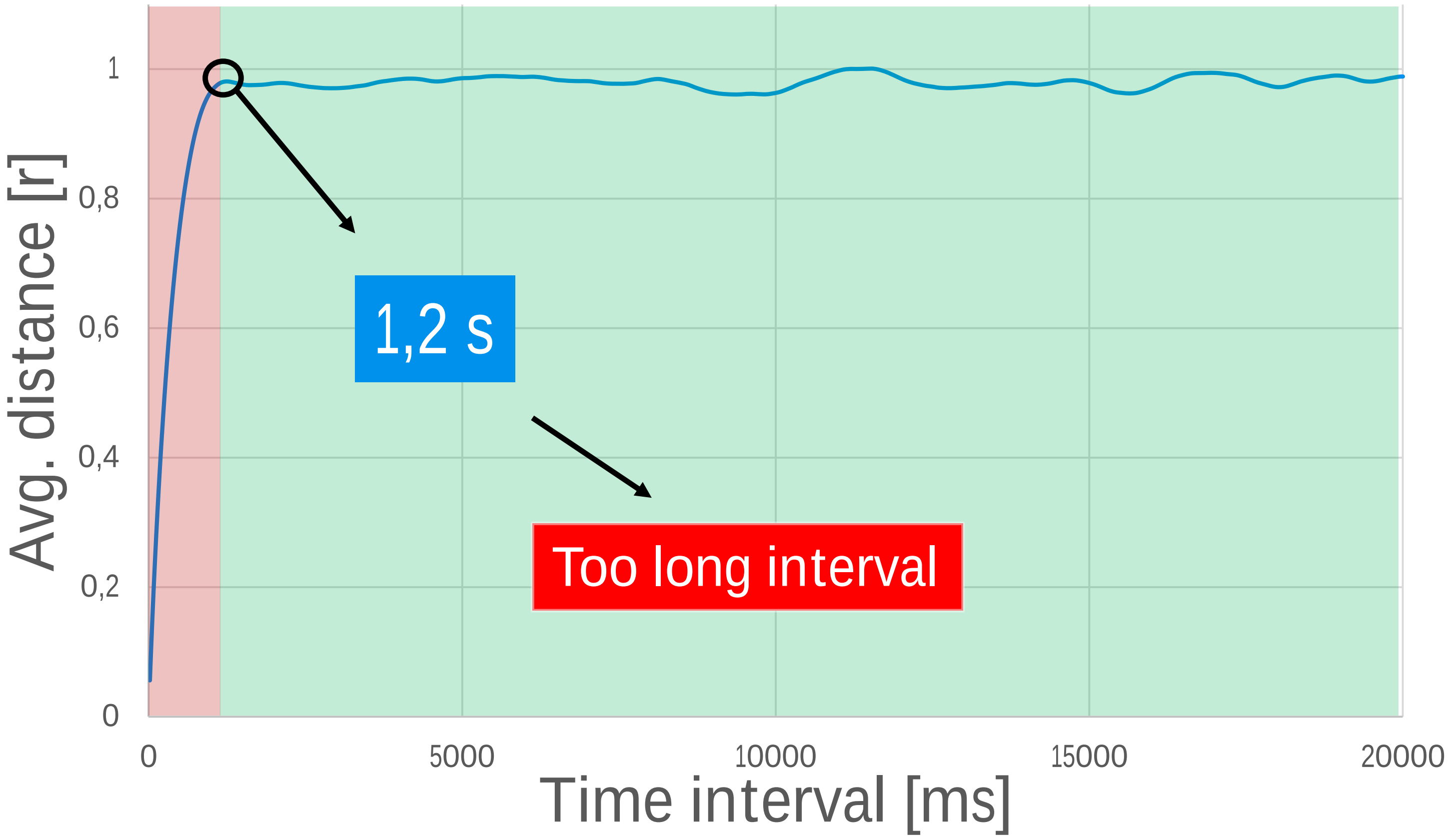
Distance



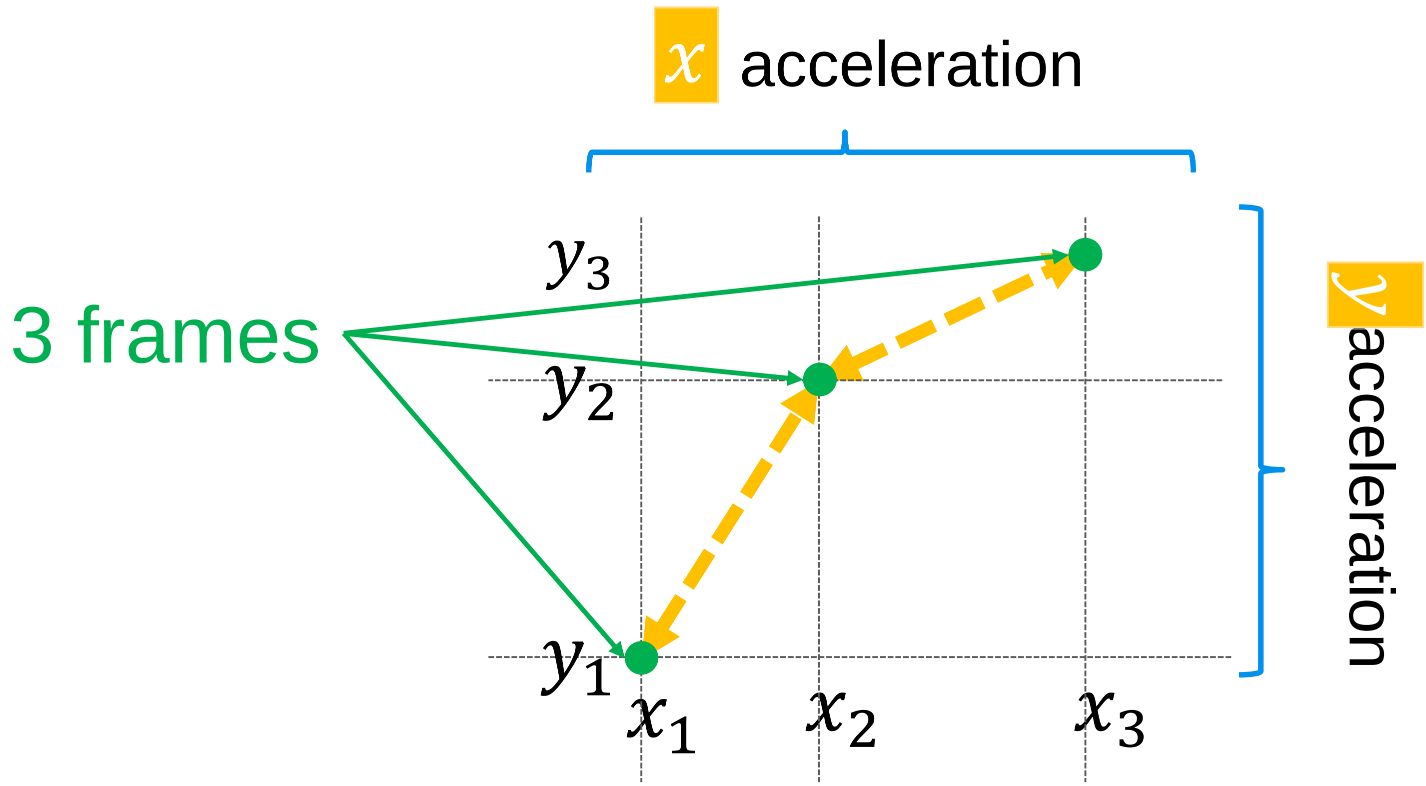
$$d = \sqrt{\Delta x^2 + \Delta y^2}$$

Correct time interval is required

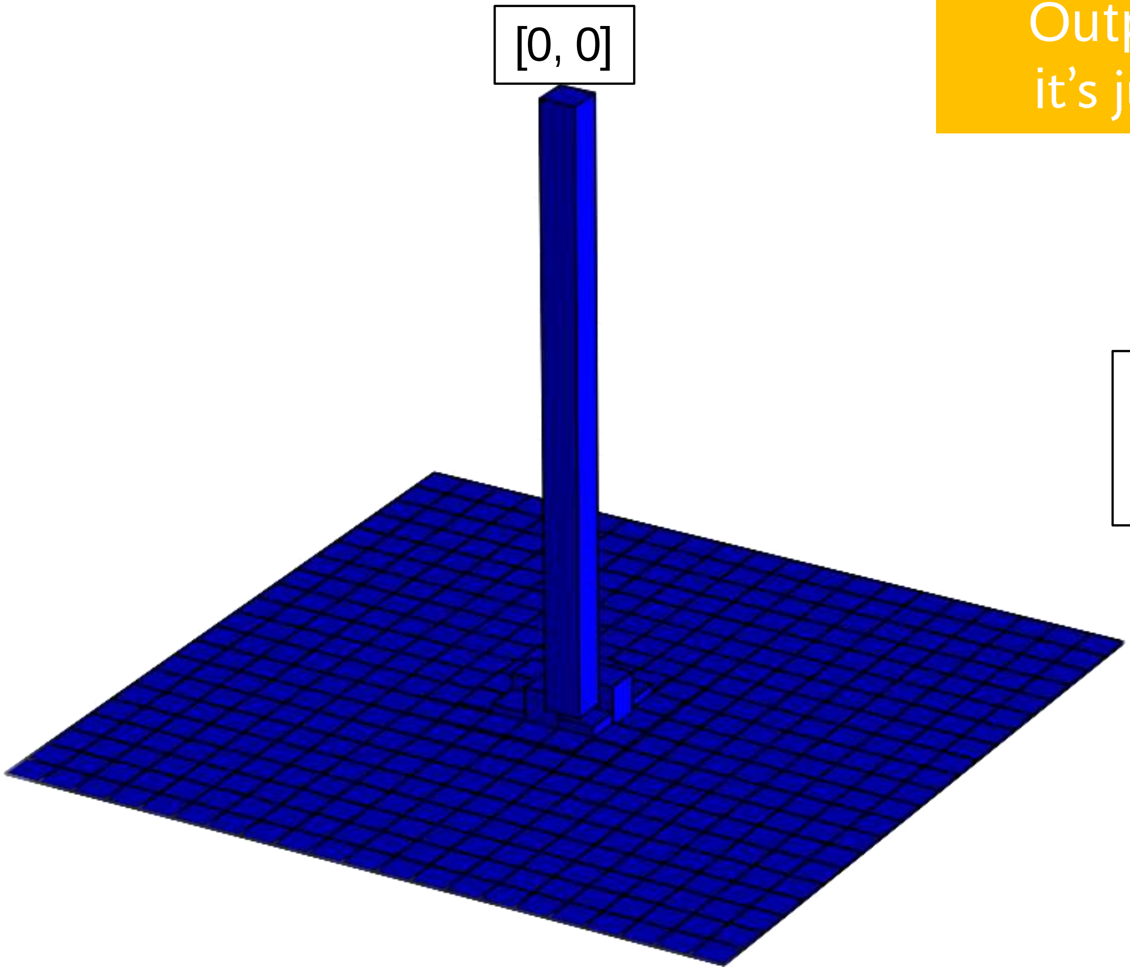
Distance Correlation time



Generating Numbers from **Acceleration**



Acceleration Histogram

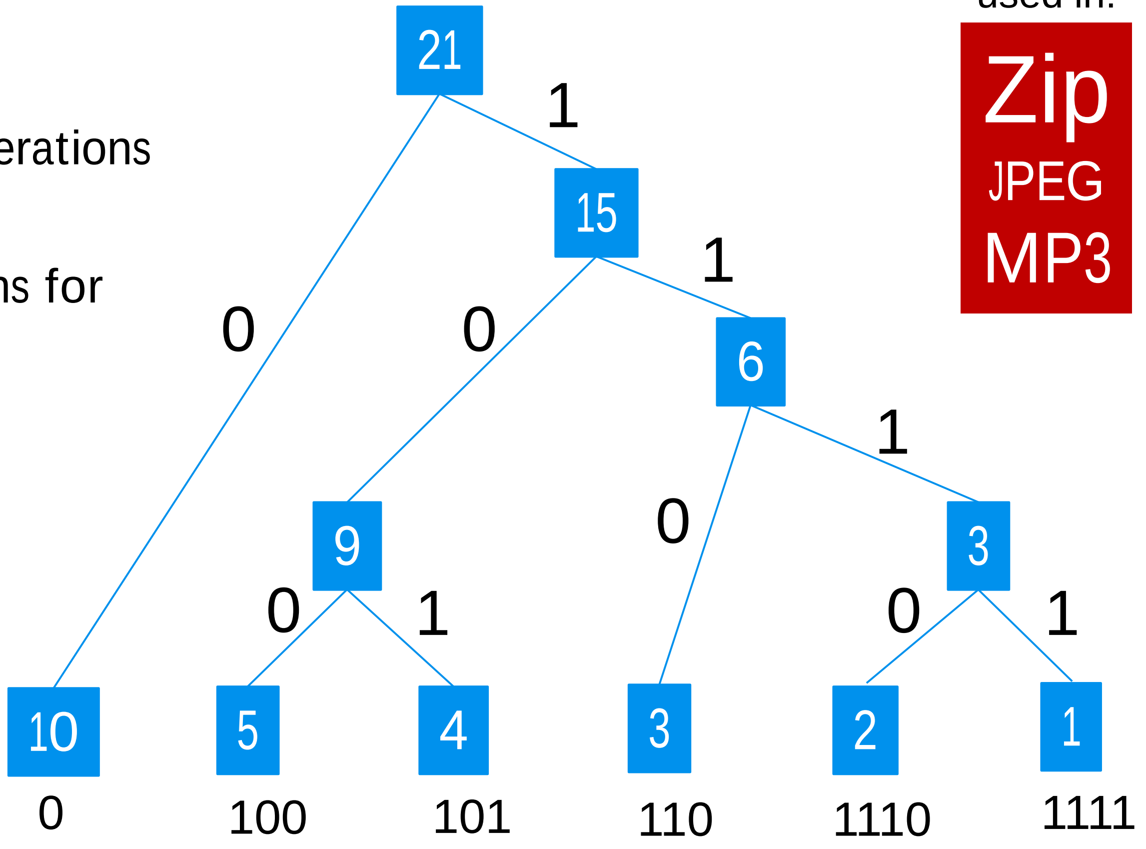


Output IS random...
it's just not uniform

We need
proper binary coding

Compression: Huffman coding

- Separate for x and y accelerations
- Codes pairs of accelerations for optimum efficiency



Currently used in:

Zip
JPEG
MP3

Compression: $L > H$

Entropy
$$H = 1.754$$

$$L = 1.762$$
Avg. length

99.55% efficiency

Imperfect coding

Problems:
- Contains patterns
- Biased

Perfect coding: $H = L$

XOR: Extractor of randomness

How it works:

x data:	1 0 1 0 1 1 1 0 0 0 0 1 1 0 1 0 …
y data:	1 0 0 1 1 1 1 1 1 0 0 1 0 0 1 1 …
output:	0 0 1 1 0 0 0 1 1 0 0 0 1 0 1 1 …

Problems:
- ~~Contains patterns~~
- Biased

Elimination of any patterns

½ size reduction

Neumann's Debiasing Algorithm

How it works:

1 1	1 0	1 1	1 0	0 1	0 1	1 1	1 1	0 0	0 0	1 1	1 0	1 0	1 0	…
	1		1	0	0						1	1	1	…

Problems:
- ~~Contains patterns~~
- ~~Biased~~

More balanced bits

¾ size reduction

Produced Output

4 hours → 1,790,880 frames → 191,368 bits

1280 x 720
100fps

STATISTICAL TESTS

Our Generator vs. RANDOM.ORG


Home Games Numbers Lists & More Drawings Web Tools Statistics Testimonials Learn More Login

RANDOM.ORG

Search RANDOM.ORG
Search

True Random Number Service

Do you own an iOS or Android device? Check out our app!

Random Integer Generator

This form allows you to generate random integers. The randomness comes from atmospheric noise, which for many purposes is better than the pseudo-random number algorithms typically used in computer programs.

Part 1: The Integers

Generate 100 random integers (maximum 10,000).

Each integer should have a value between 1 and 100 (both inclusive; limits ±1,000,000,000).

Format in 5 column(s).

Part 2: Go!

Be patient! It may take a little while to generate your numbers...

Get Numbers Reset Form Switch to Advanced Mode

Note: The numbers generated with this form will be picked independently of each other (like rolls of a die) and may therefore contain duplicates. There is also the Sequence Generator, which generates randomized sequences (like raffle tickets drawn from a hat) and where each number can only occur once.

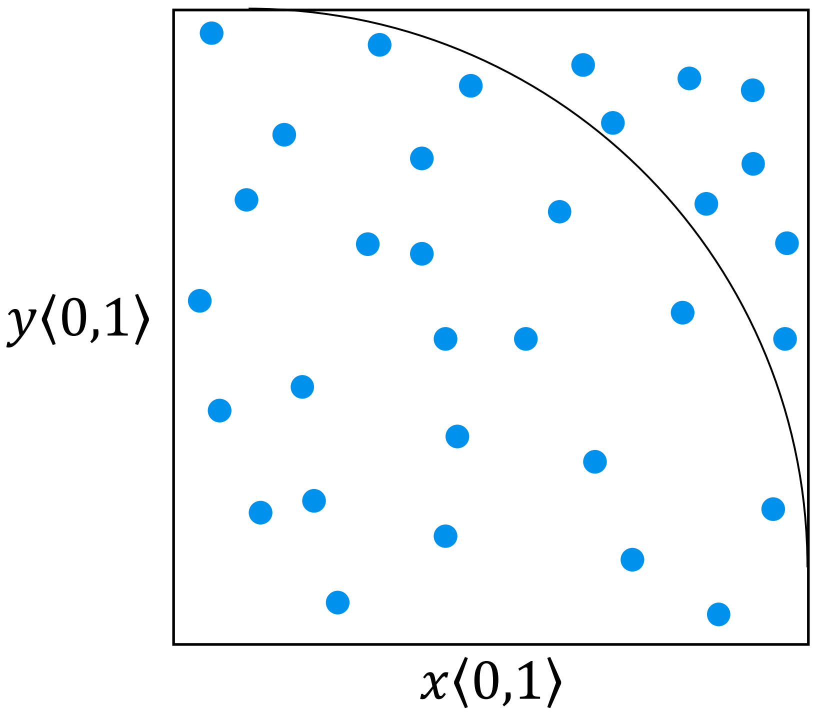
Follow @RandomOrg 3,526 followers
Like Share 405K
G+1 19k

© 1998-2016 RANDOM.ORG
Terms and Conditions
About Us

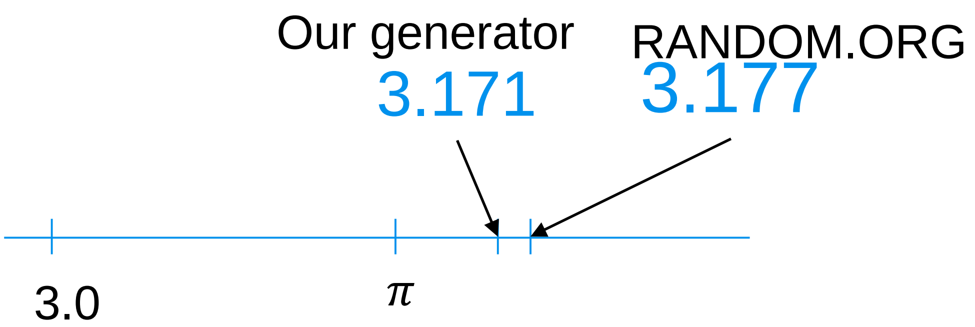
Largest *TRNG* on the internet

Atmospheric noise

Monte Carlo Computation: π



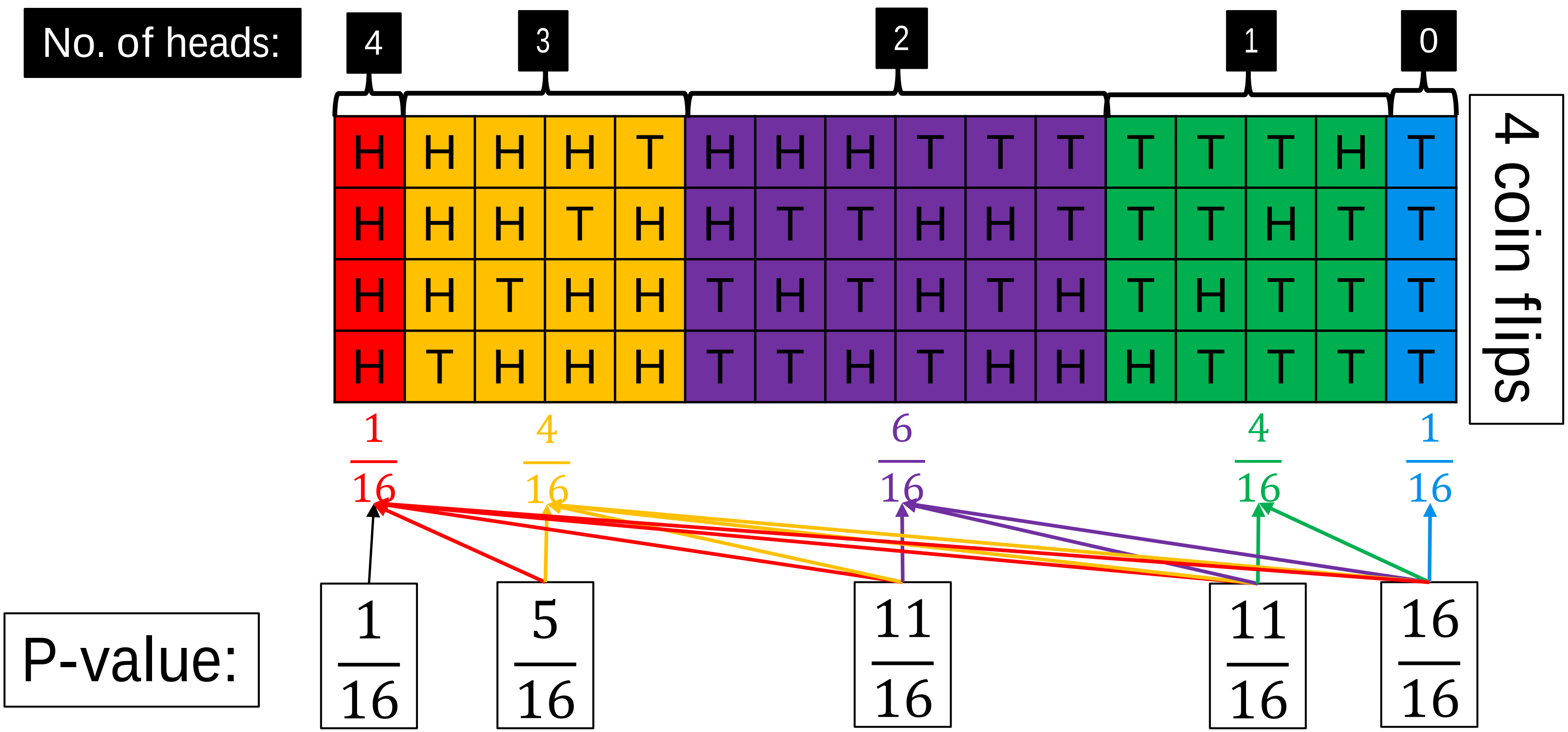
$$x^2 + y^2 \leq 1$$



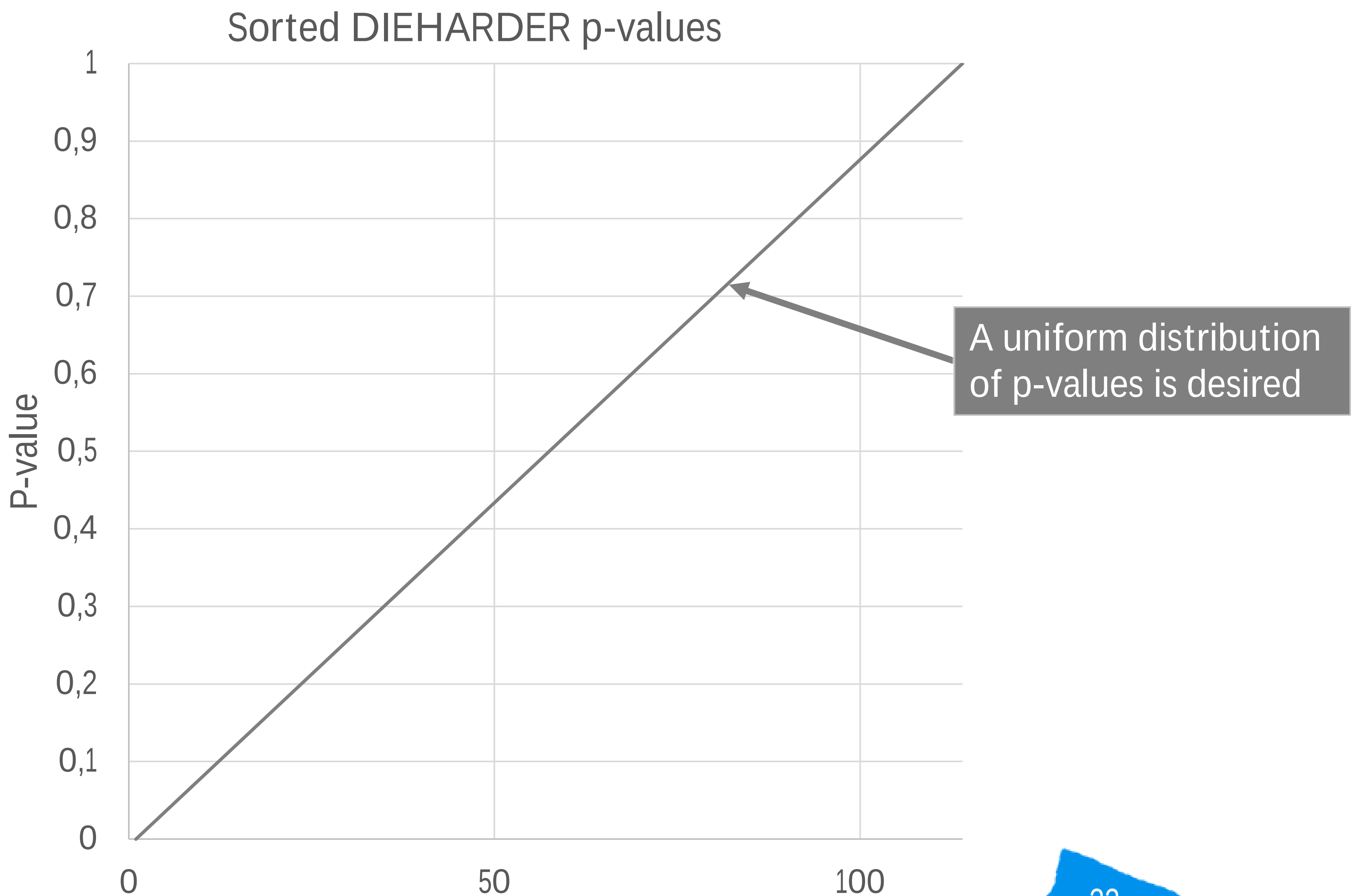
DIEHARDER

- Better parameterized (and expanded) DIEHARD tests
- Battery of 114 statistical test on RNG
- Output of each test: p-values based on the assumption of a perfect TRNG

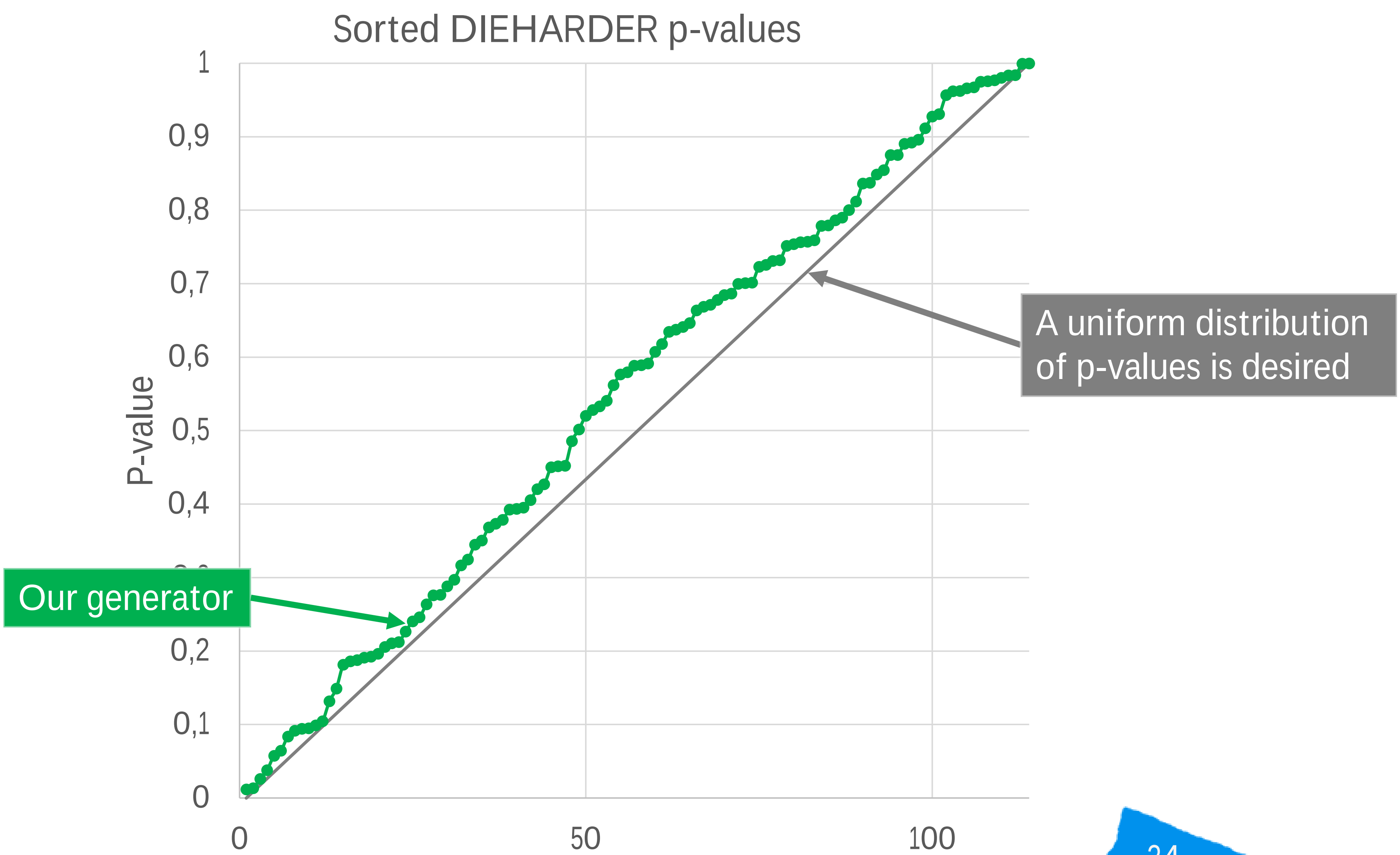
What are p-values? – Example: No. of Heads



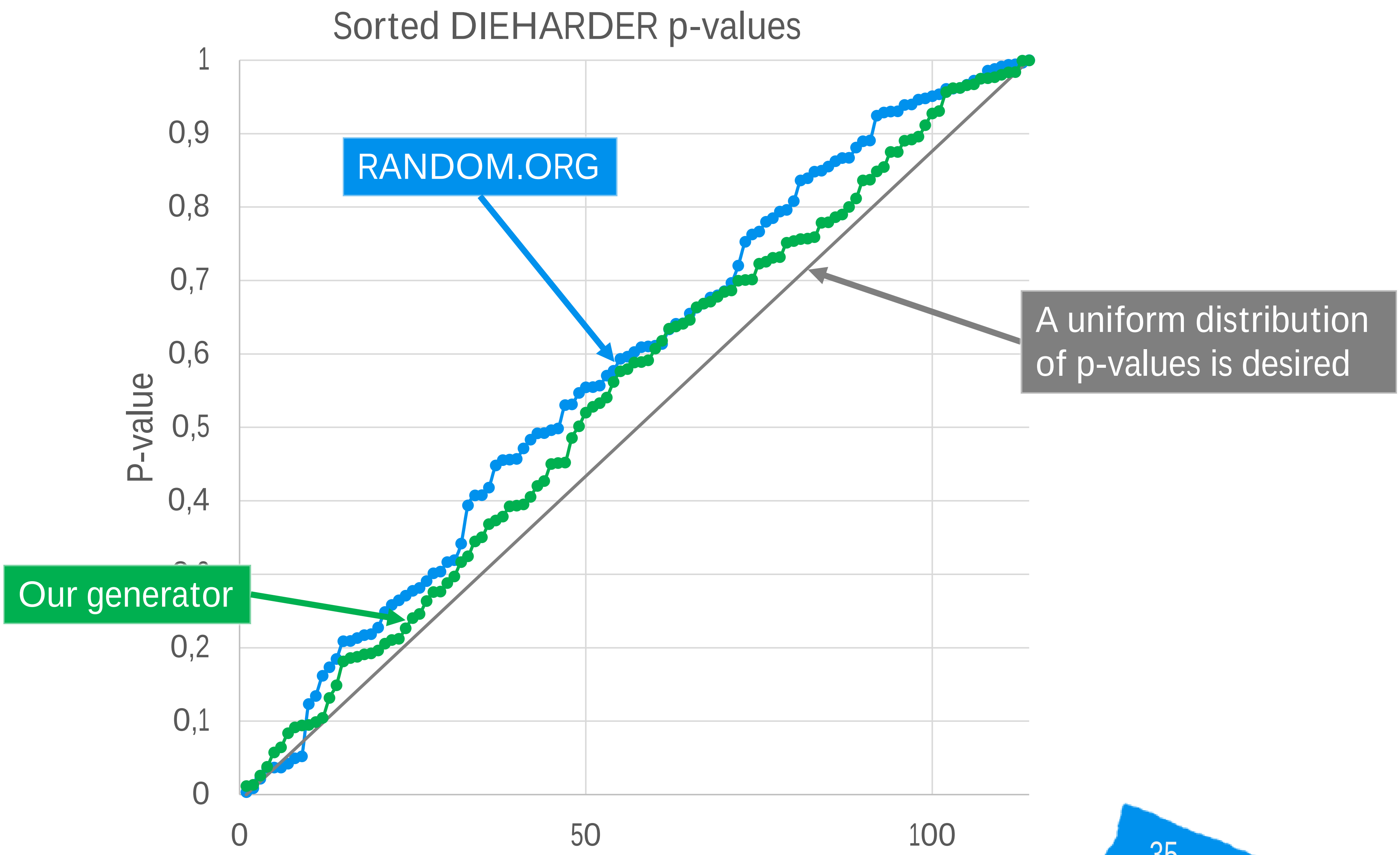
DIEHARDER: p-value Distribution



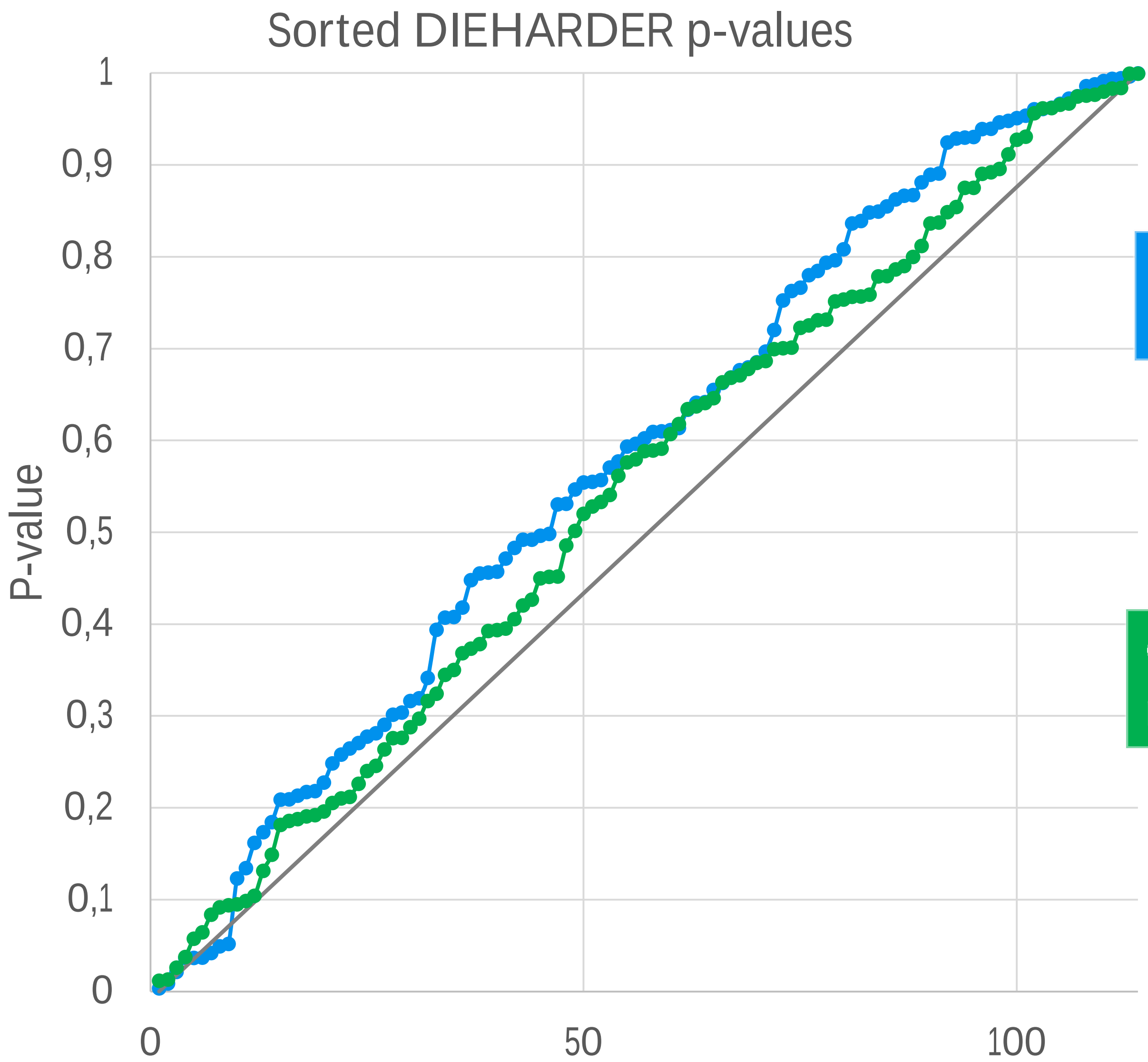
DIEHARDER: p-value Distribution



DIEHARDER: p-value Distribution



DIEHARDER: p-value Distribution

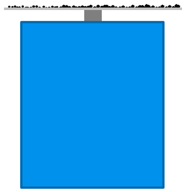


RANDOM.ORG's mean difference: 0.0807

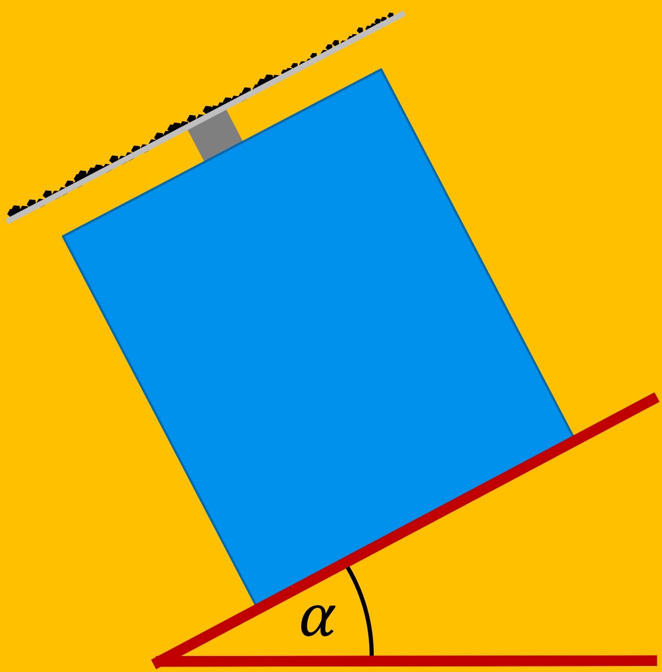
Our generator's mean difference: 0.0474

Problem definition

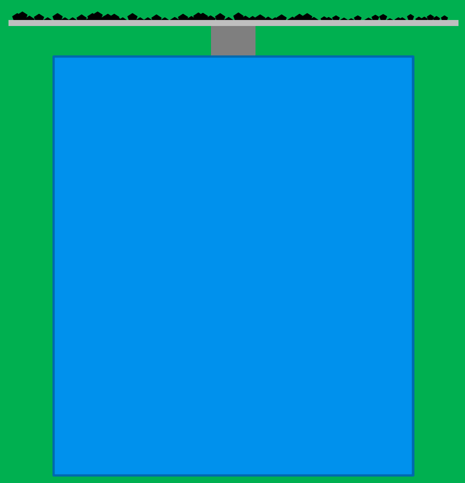
"Truly random numbers are a very valuable and rare resource. Design, produce, and test a mechanical device for producing random numbers. Analyse to what extent the randomness produced is safe against tampering."



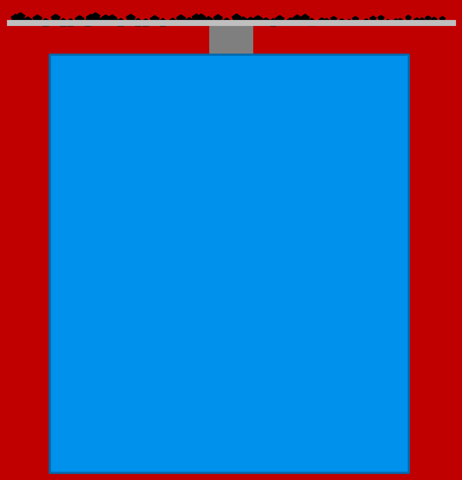
Angle



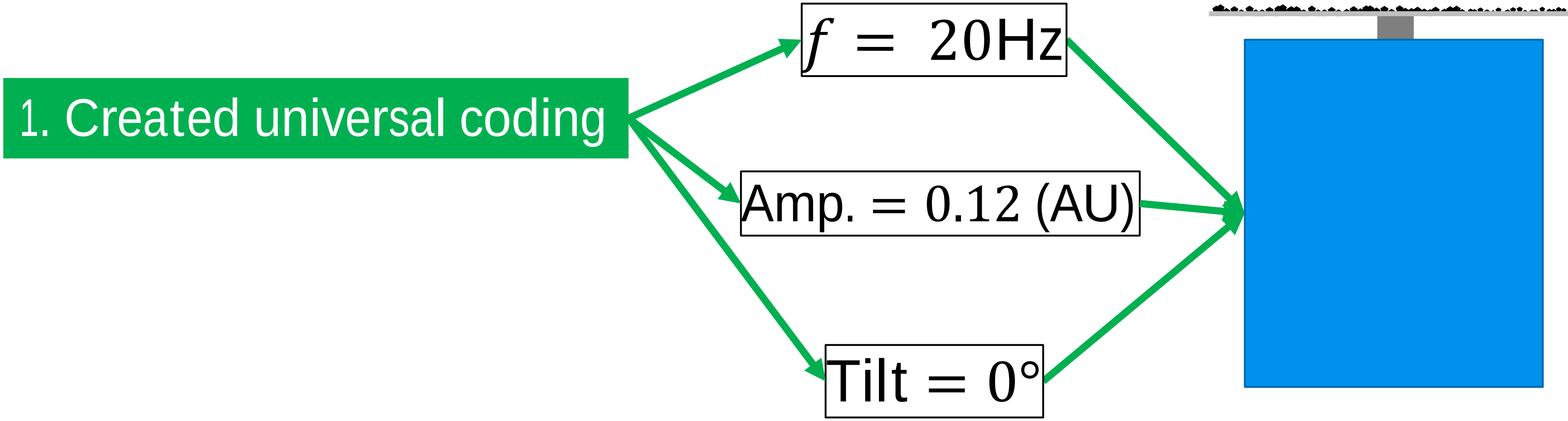
Amplitude



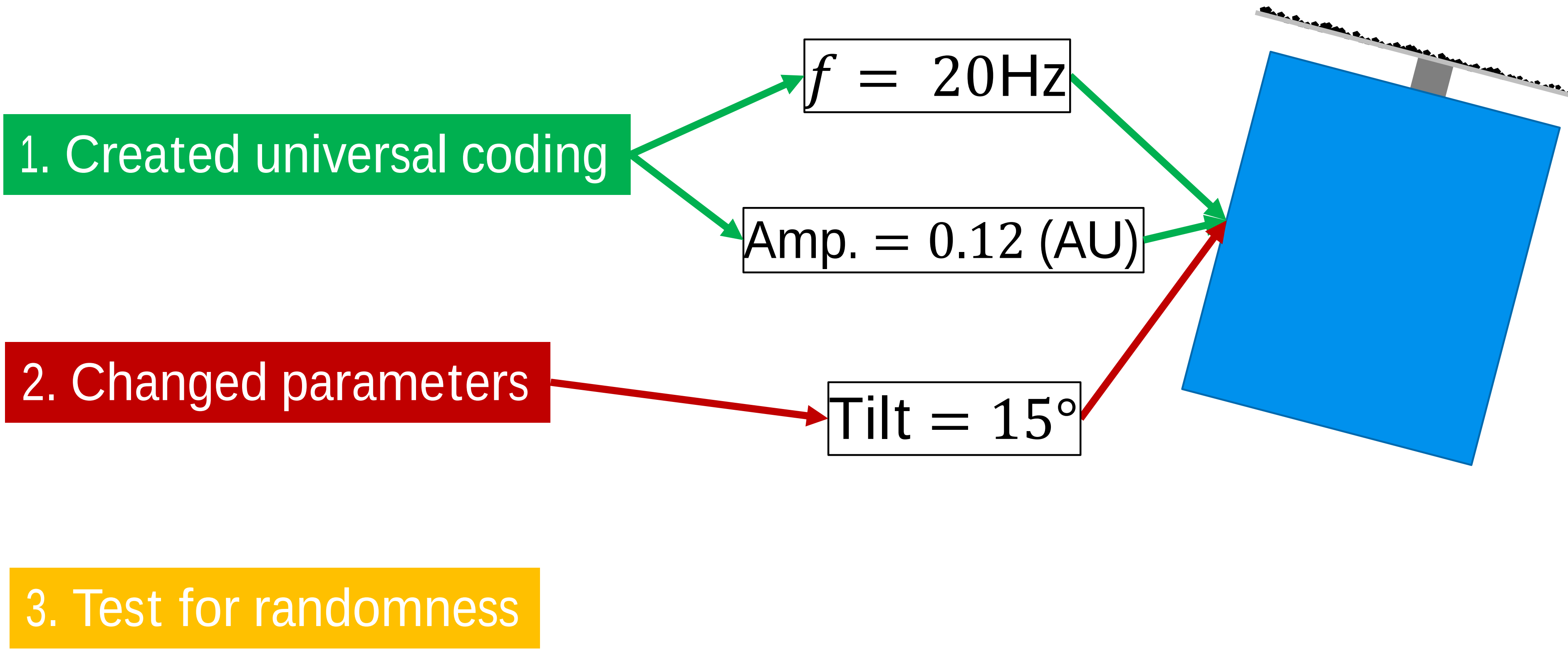
Frequency



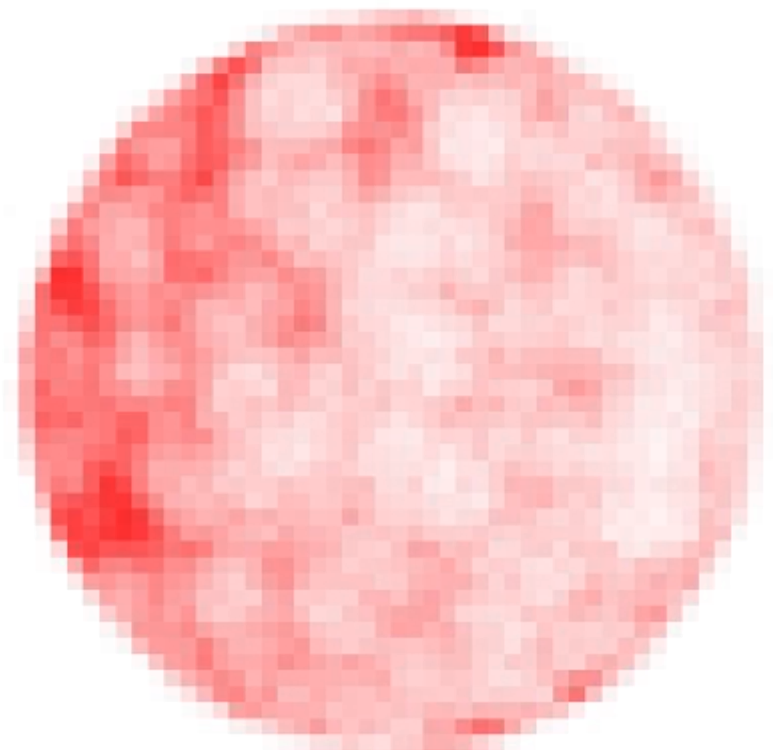
How we test against tampering



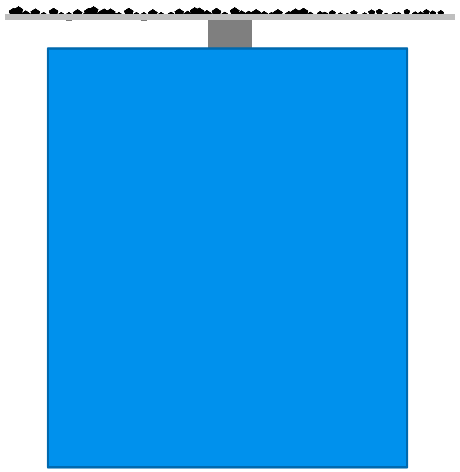
How we test against tampering



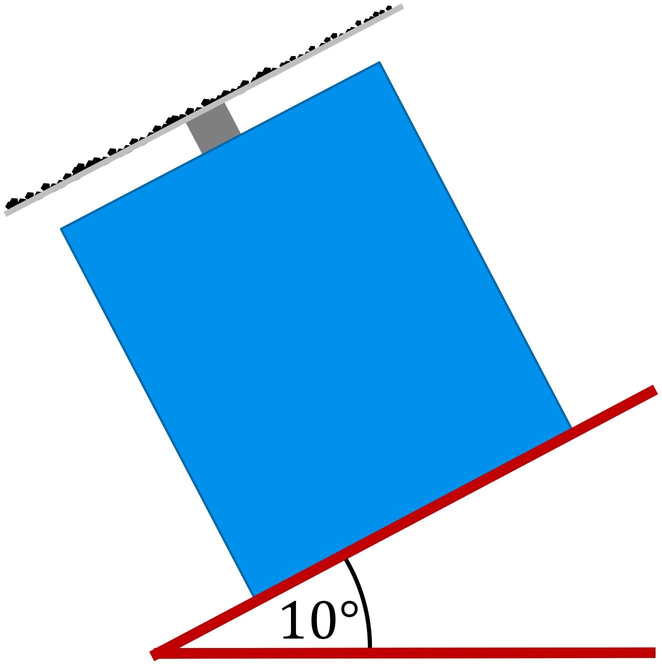
Normal settings: heat map



Near-uniform distribution



15° tilt



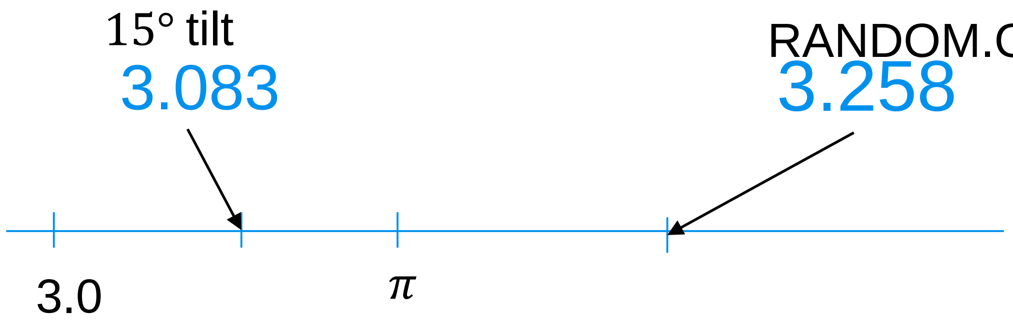
Heat map:

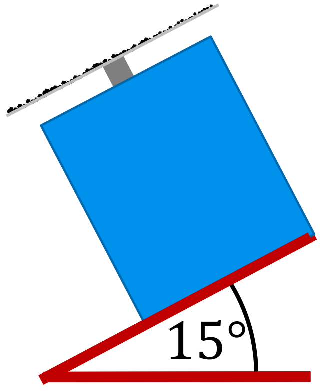
Tilt can easily be observed

Bud how good is the data anyway?

15° tilt: output quality

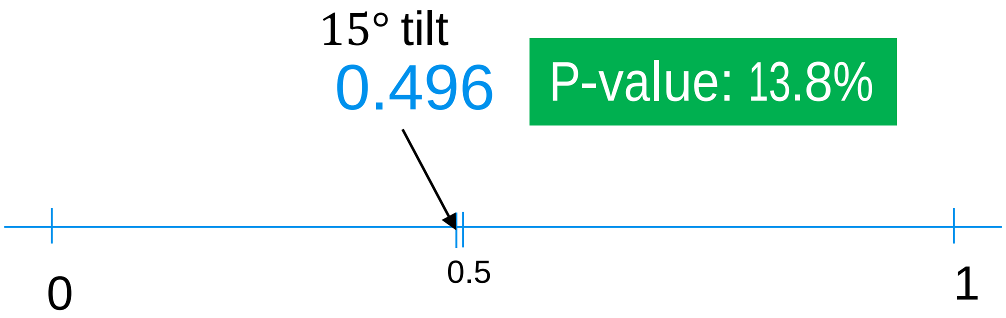
Mote Carlo: π



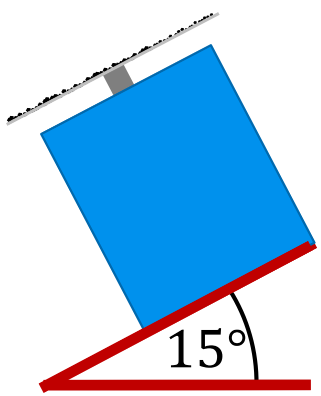


15° tilt: output quality

Avg. bit value

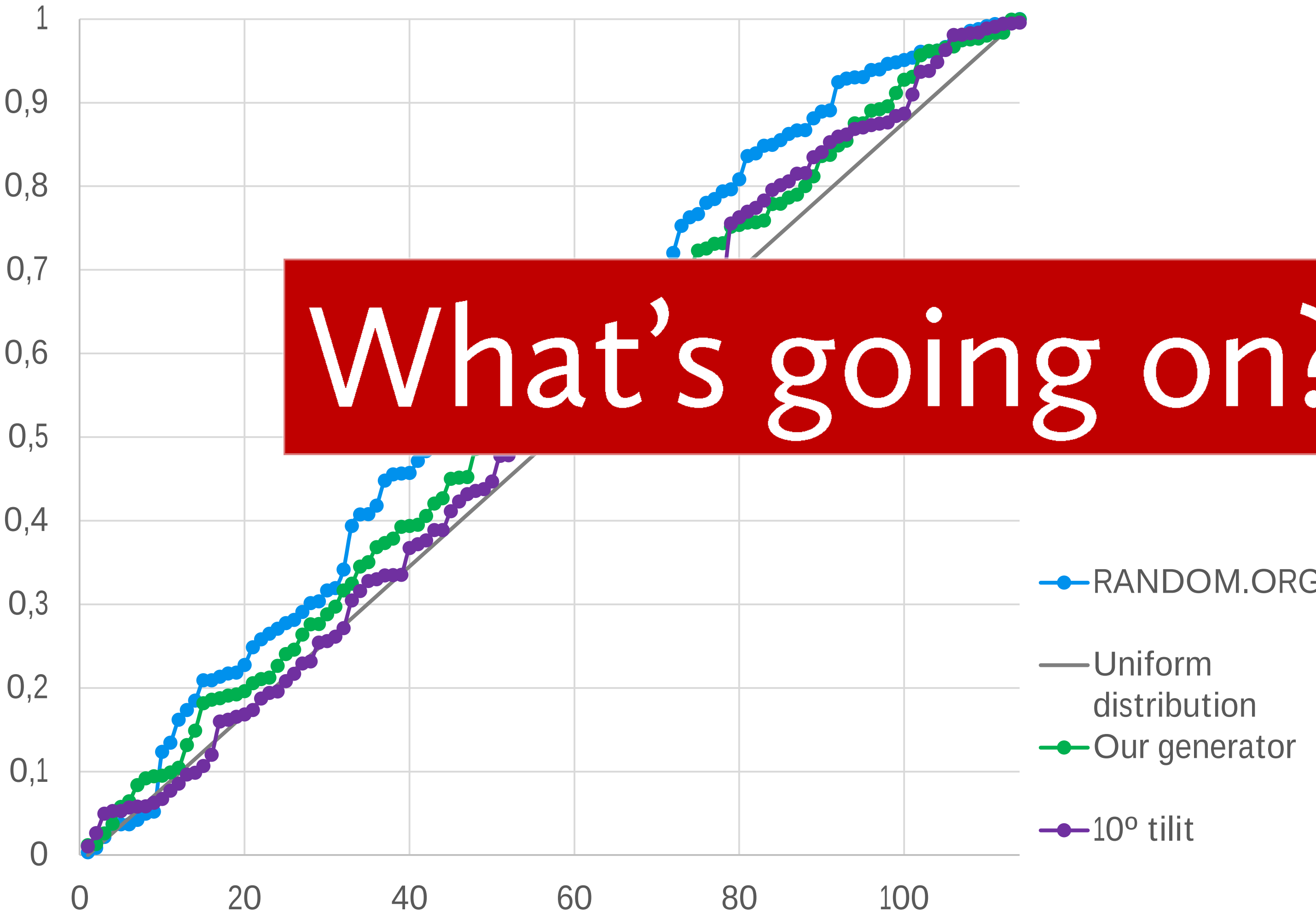


P-value: 13.8%

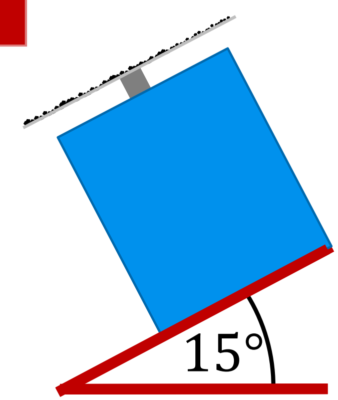


15° tilt: output quality

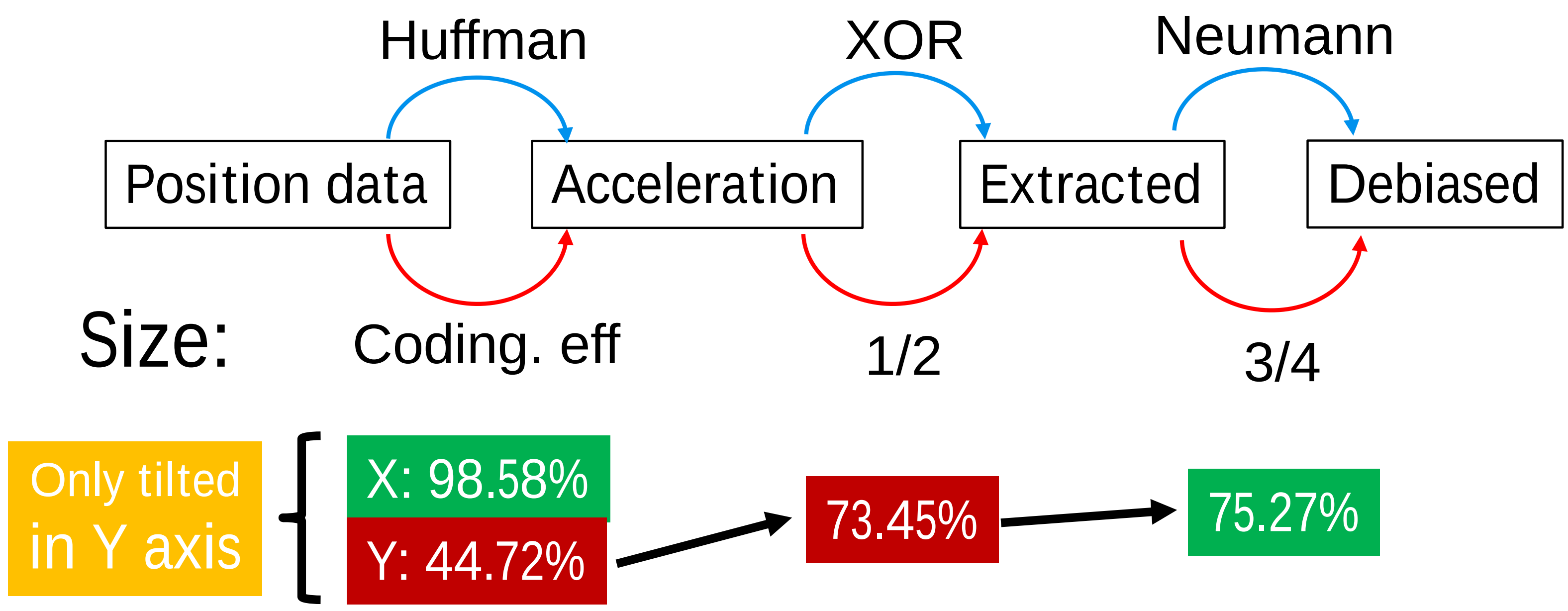
DIEHARDER



What's going on?



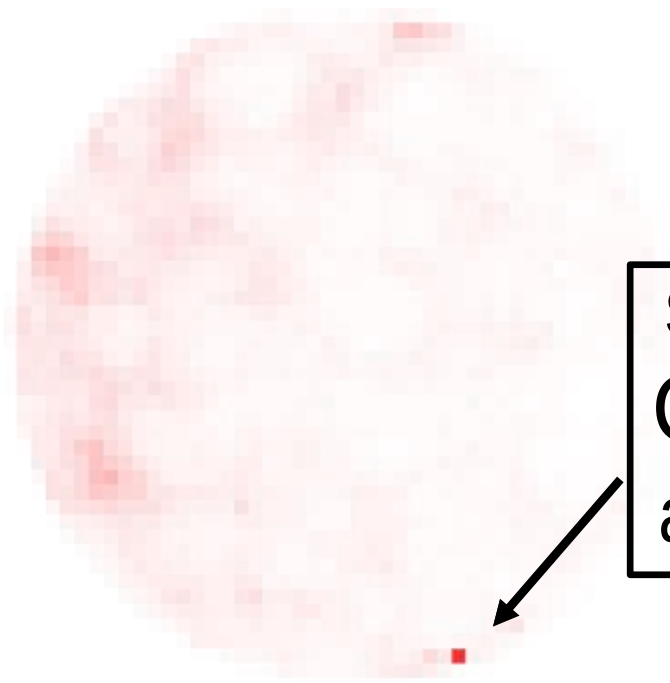
15° tilt: output size



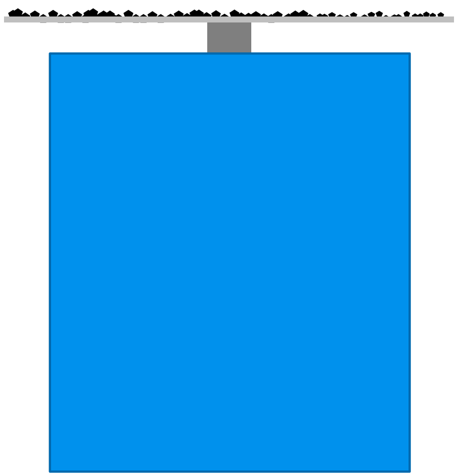
Difference between X and Y length
means throwing away excess data

10% lower Amp.

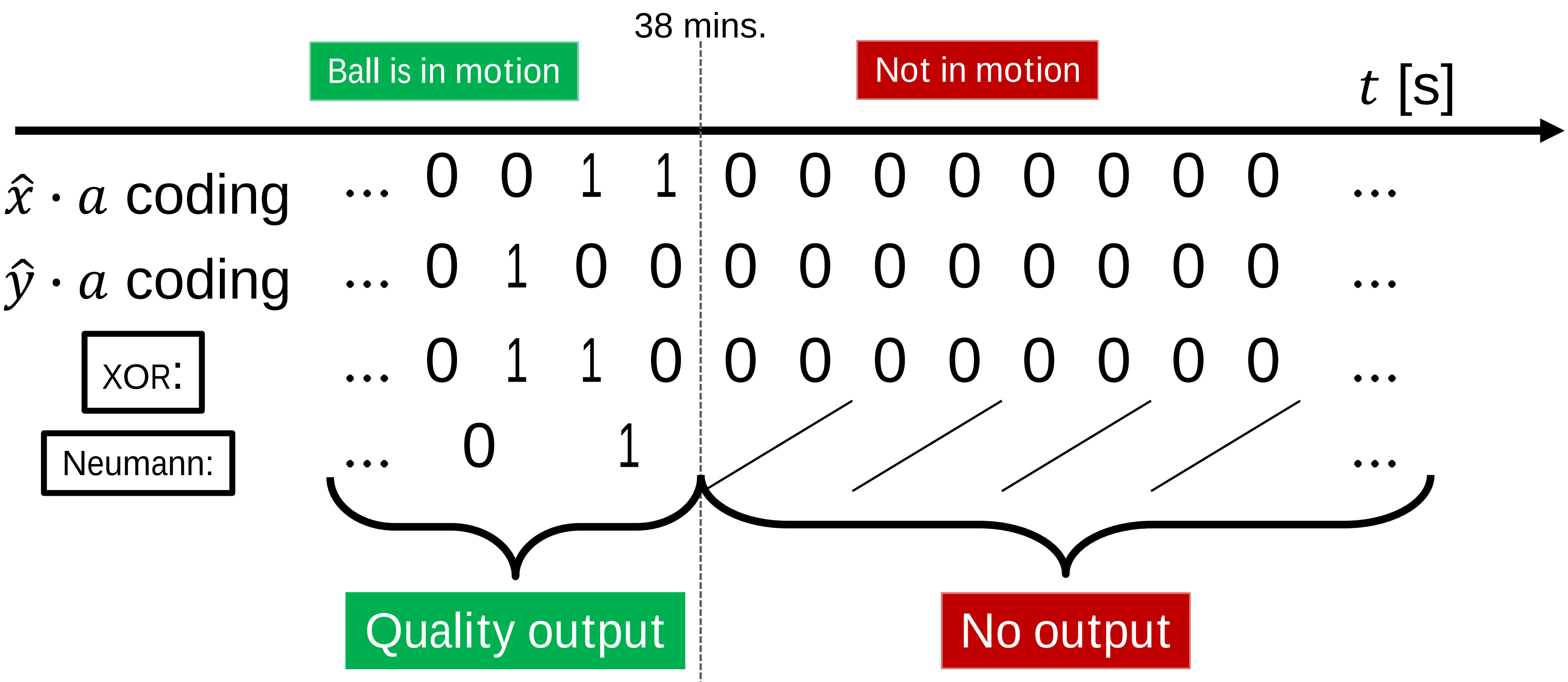
Heat map:



Stopped moving
Completely
after 38 mins.

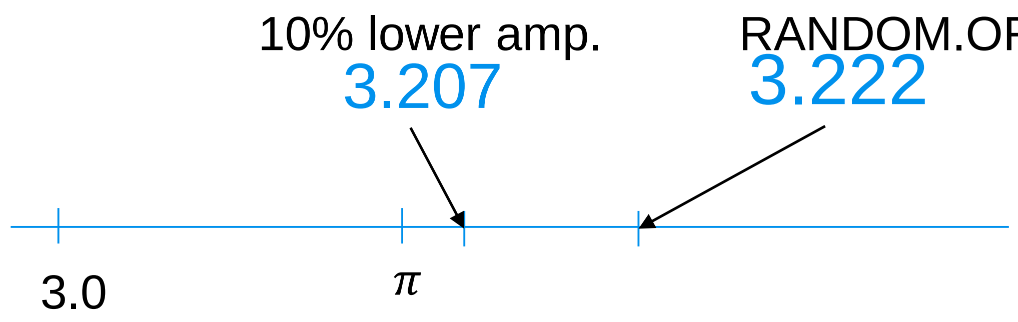


10% lower Amp.


38 mins.
Ball is in motion
Not in motion
t [s]
x̂ · a coding … 0 0 1 1 0 0 0 0 0 0 0 0 …
ŷ · a coding … 0 1 0 0 0 0 0 0 0 0 0 0 …
XOR: … 0 1 1 0 0 0 0 0 0 0 0 0 …
Neumann: … 0 1 …
Quality output
No output

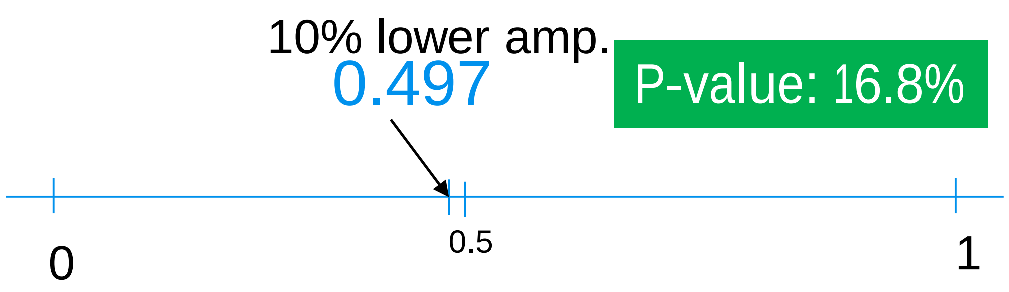
10% lower Amp.: output quality

Mote Carlo: π

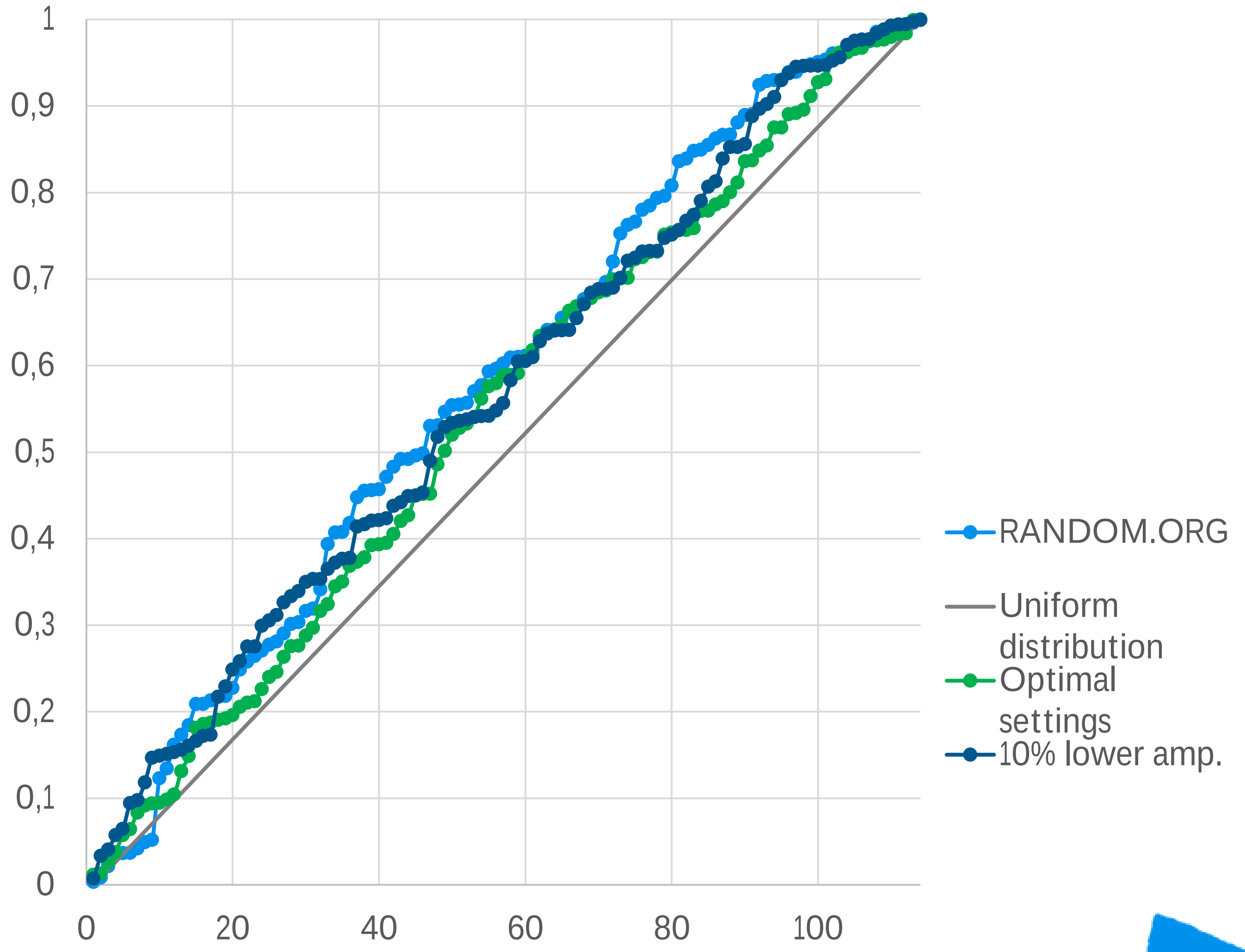


10% lower Amp.: output quality

Avg. bit value



10% lower Amp.: DIEHARDER



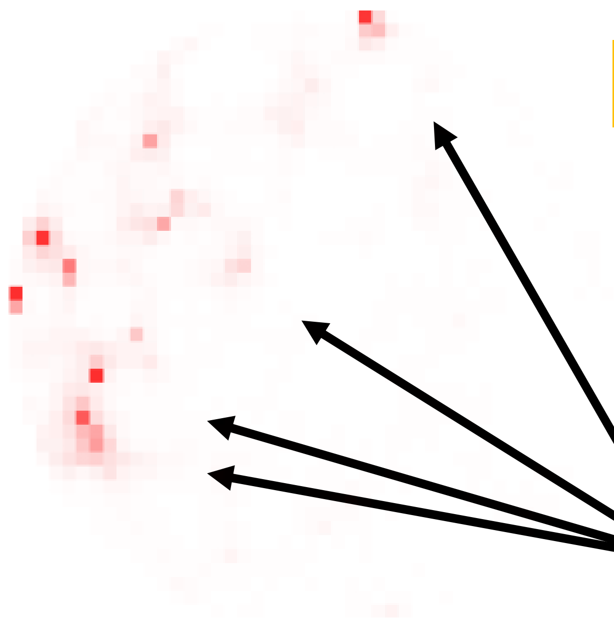
CHANGING FREQUENCY

f ⟷ analogous ⟷ Amp.

We are only interested in the *acceleration* of the crushed stone →

$$a = f^2 A$$

Heat map:

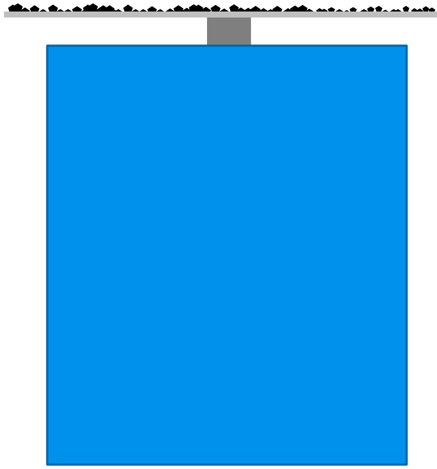


25% increase in f

Increase in f lowered the speaker's amplitude

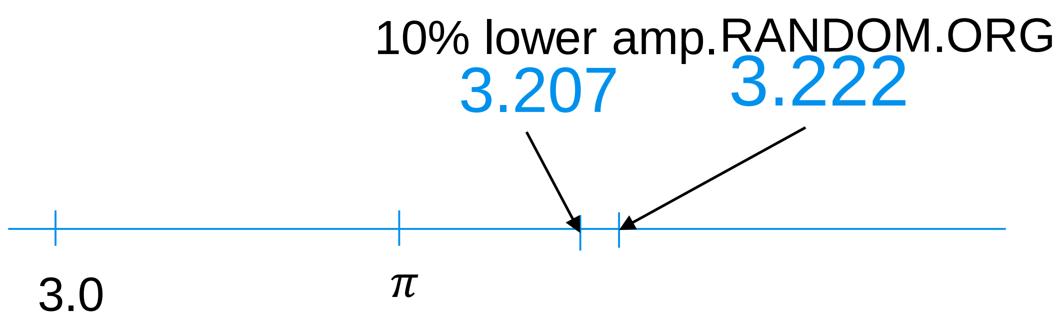
Lower Af^2

Couldn't easily escape from these holes



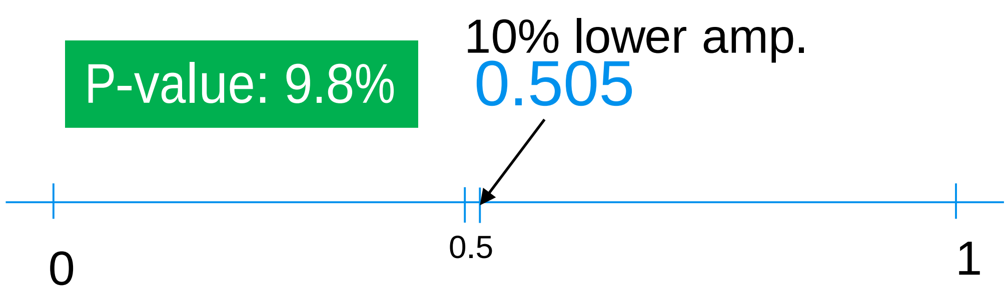
25% increase in f: output quality

Mote Carlo: π

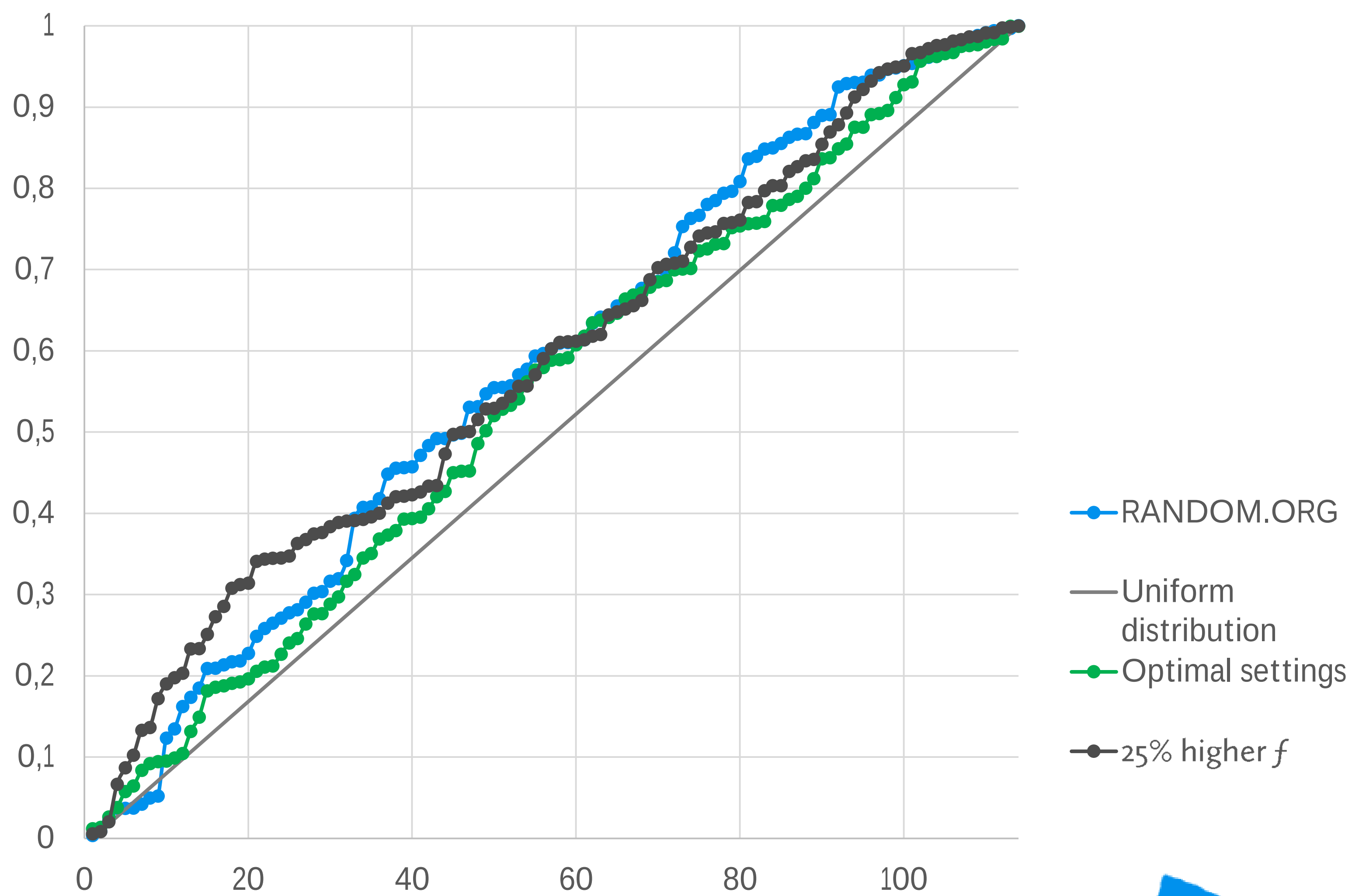


25% increase in f: output quality

Avg. bit value



25% increase in f: DIEHARDER



Conclusion

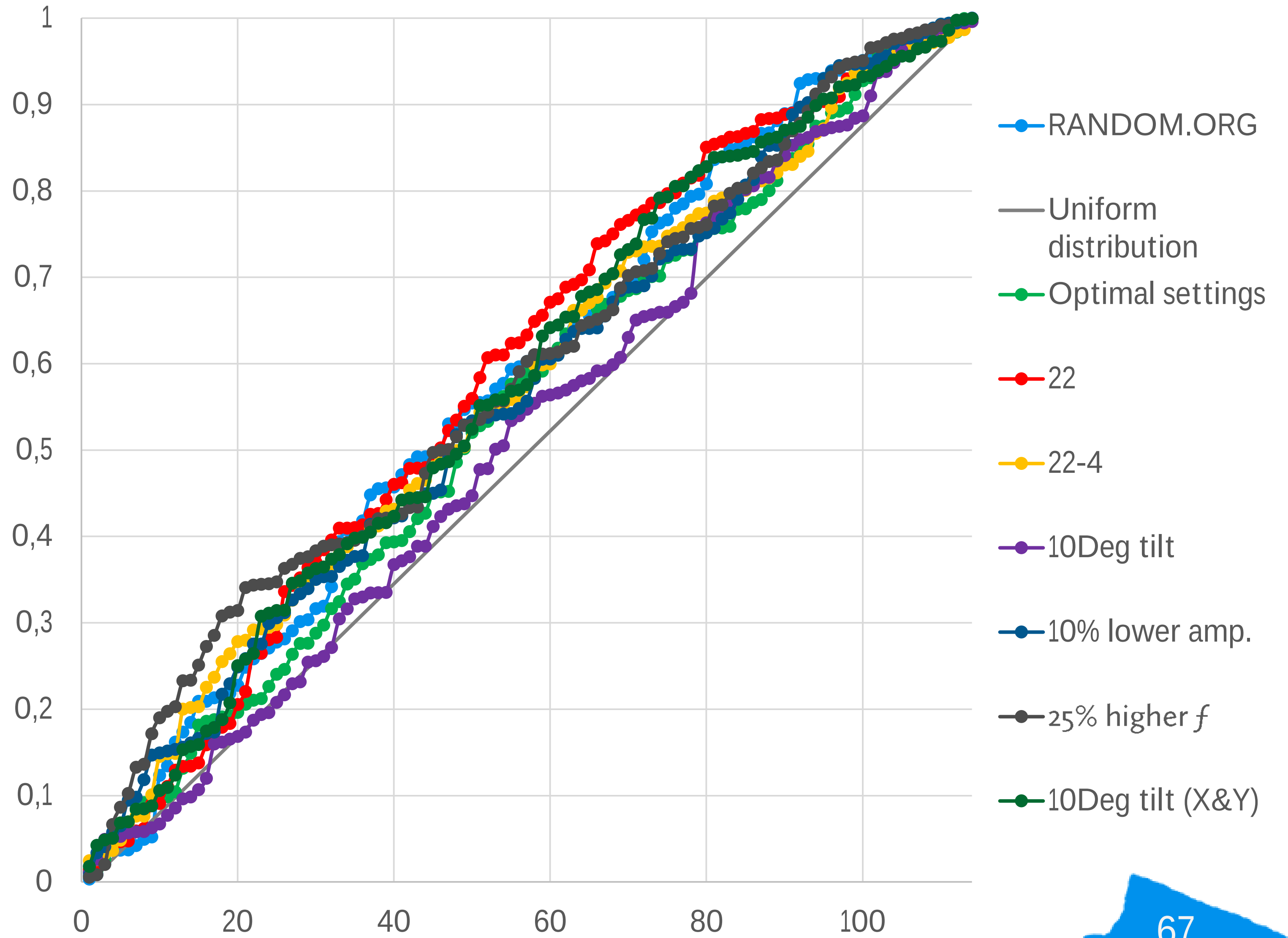
1. Created a mechanical device + processing software (automated tracking, all algorithms, heat maps, position data,...)

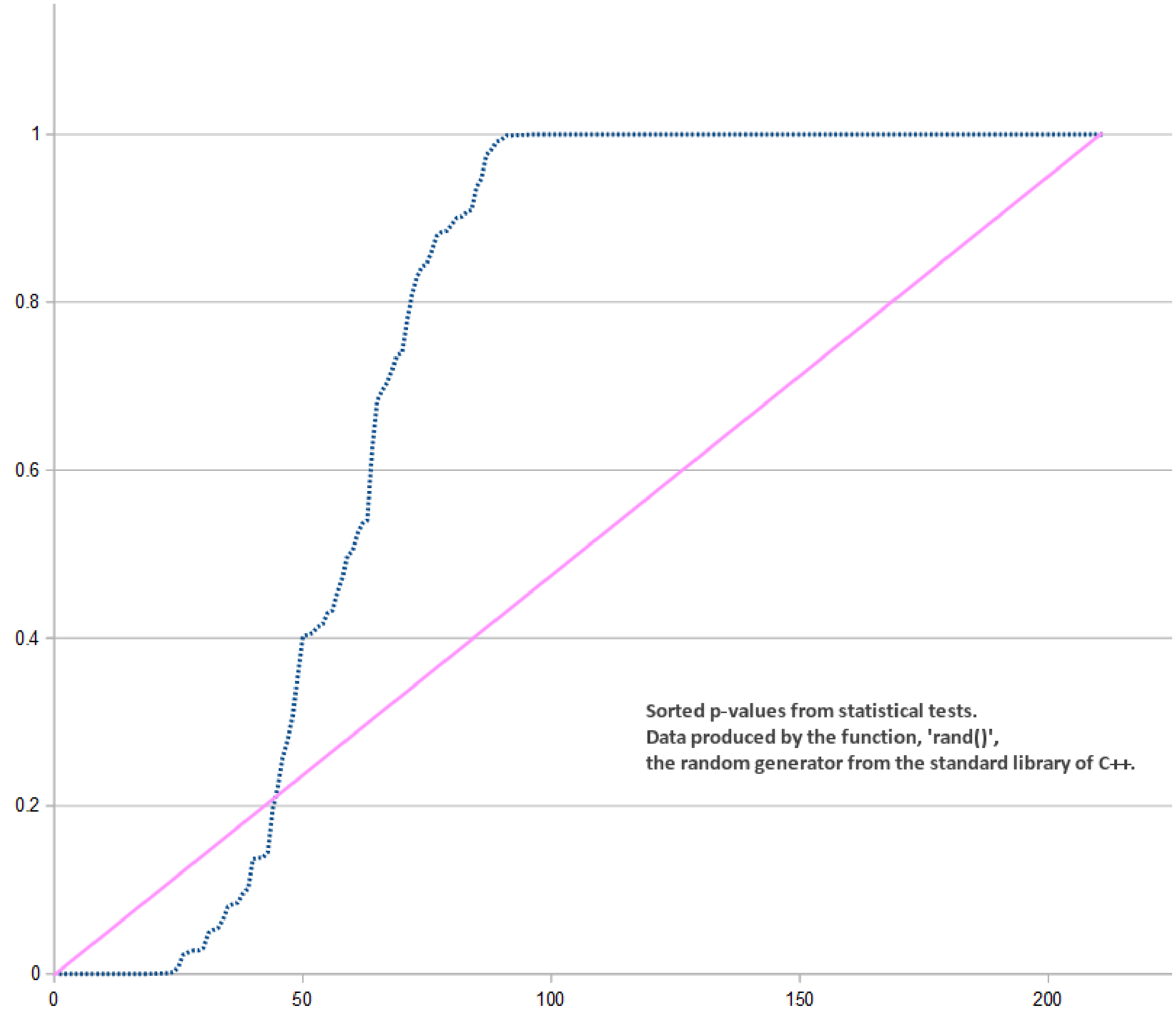
2. Analyzed the randomness: **Extremely high**

3. Analyzed safeness against tampering: **Extremely safe**

APPENDIX

All DIEHARDER tests




1
0.8
0.6
0.4
0.2
0
0
50
100
150
200
Sorted p-values from statistical tests.
Data produced by the function, 'rand()',
the random generator from the standard library of C++.

Acceleration Data

-4	1	00	00
-4	1	01	00
0	0	02	04
0	0	03	0-1
-1	0	00	0-1
-1	0	-37	03
0	-1	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
1	-2	0	0
0	0	5	6
0	0	0	0
0	0	0	0
0	0	-2	-2
		0	0
		0	0
		0	0
		0	0
		-3	-5
		0	-1

$$\sout{x, y} \longrightarrow \sout{\dot{x}, \dot{y}} \longrightarrow \ddot{x}, \ddot{y}$$

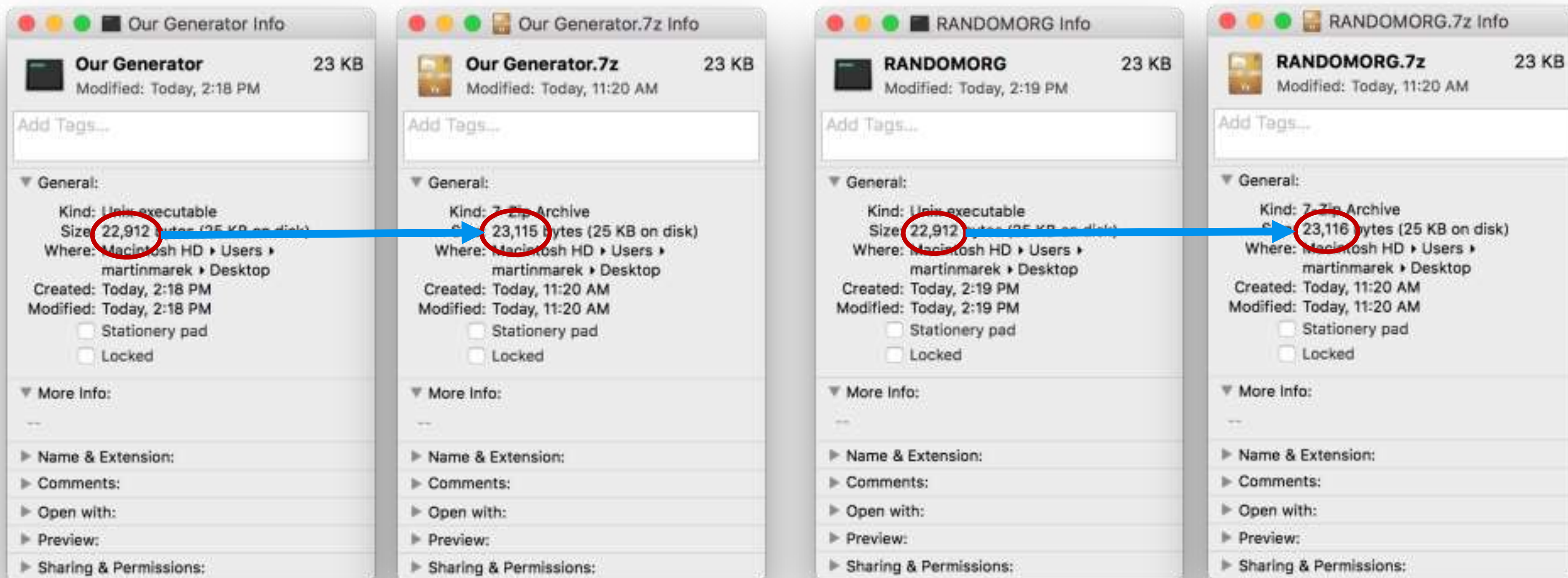
Contionous | Contionous | Discrete

Imbalanced

Contionous → Too long time interval

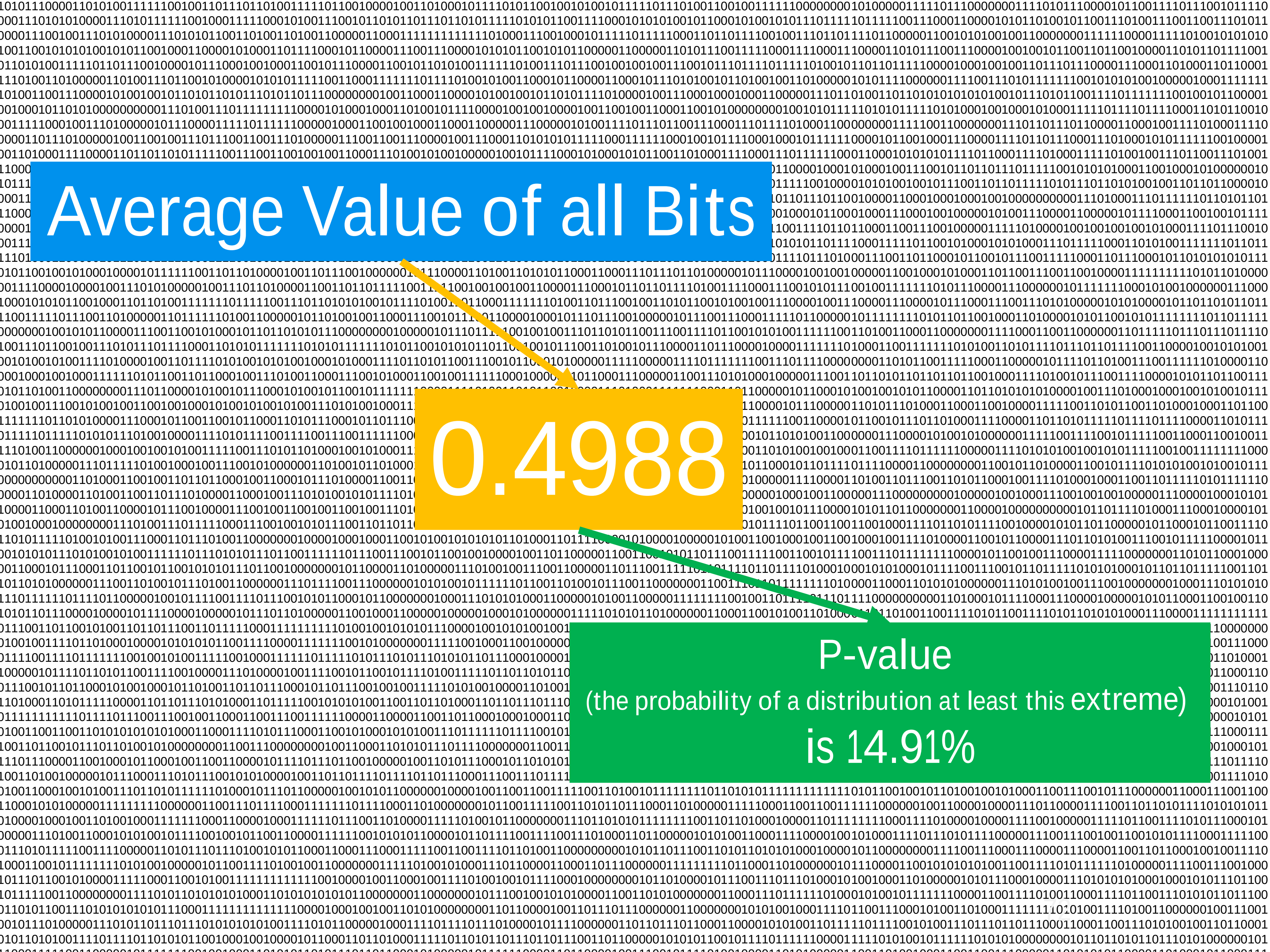
Discrete → most data

LZMA2 Compression



22,912 → 23,115

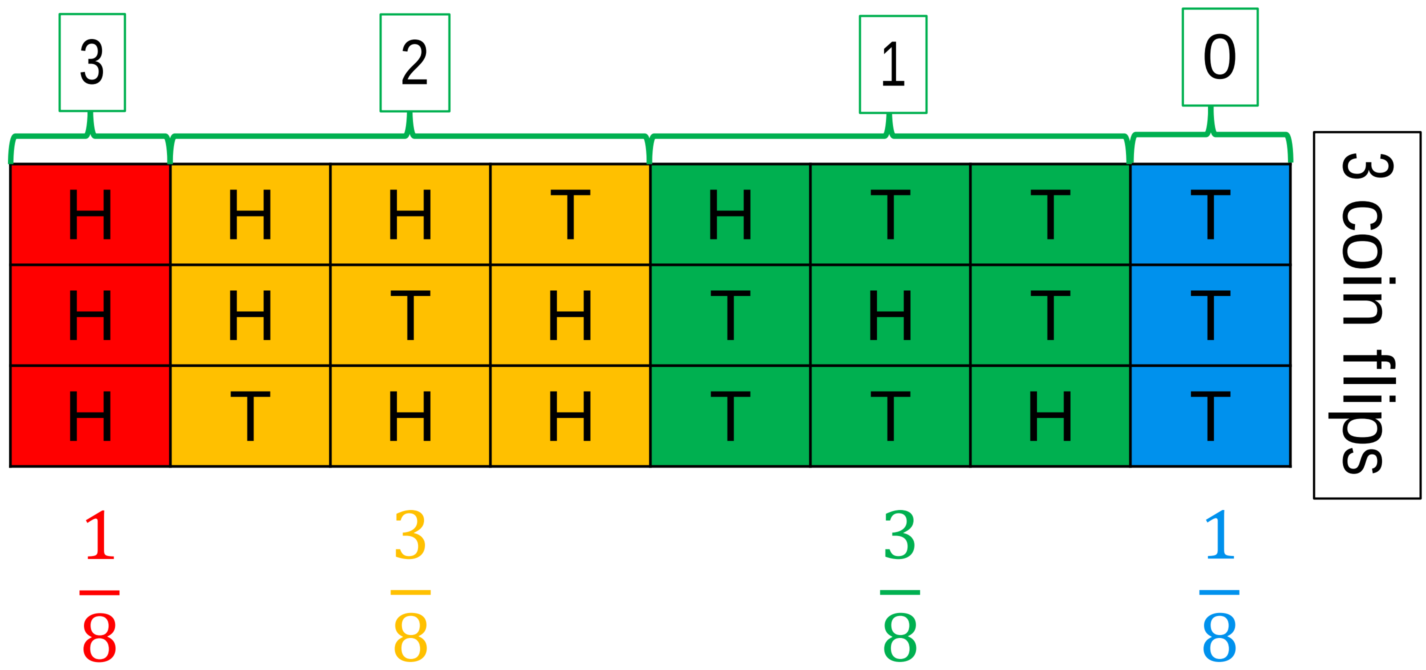
22,921 → 23,116


Average Value of all Bits
0.4988
P-value
(the probability of a distribution at least this extreme)
is 14.91%

DIEHARDER

- Battery of 114 statistical test on RNG
- Better parameterized DIEHARD tests
  - Expanded by National Institute for Standards and Technology's tests
- Output of each test: p-values compared to a perfect TRNG

What are p-values? – Example: No. of Heads



Acceleration	Coding
0	01010
1	01011
-1	01001
-2	01000
2	01100
-3	00111
3	01101
-4	00110
-5	00101
4	01111
-6	00100
5	10000
6	10001
-7	00011
-8	00010
7	10010
-9	00001
8	10011
9	10100
-10	00000
10	10101

Compression

2.75x size reduction

Acceleration	Coding	Probab.
0	1	72.06%
1	011	9.07%
-1	000	5.31%
-2	0011	2.79%
2	01011	2.29%
-3	01000	1.66%
3	00100	1.13%
-4	010100	1.03%
-5	010010	0.77%
4	001011	0.74%
-6	0101010	0.48%
5	0100111	0.47%
6	0010101	0.35%
-7	0010100	0.34%
-8	01001101	0.22%
7	01001100	0.22%
-9	010101110	0.16%
8	010101101	0.15%
9	010101100	0.11%
-10	0101011111	0.10%
10	0101011110	0.06%

Std. coding → 5 bit.

Comp. → 1.82 bit.

Acc.	Coding
0	1
1	011
-1	000
-2	0011
2	01011
-3	01000
3	00100
-4	010100
-5	010010
4	001011
-6	0101010
5	0100111
6	0010101
-7	0010100
-8	01001101
7	01001100
-9	010101110
8	010101101
9	010101100
-10	0101011111
10	0101011110

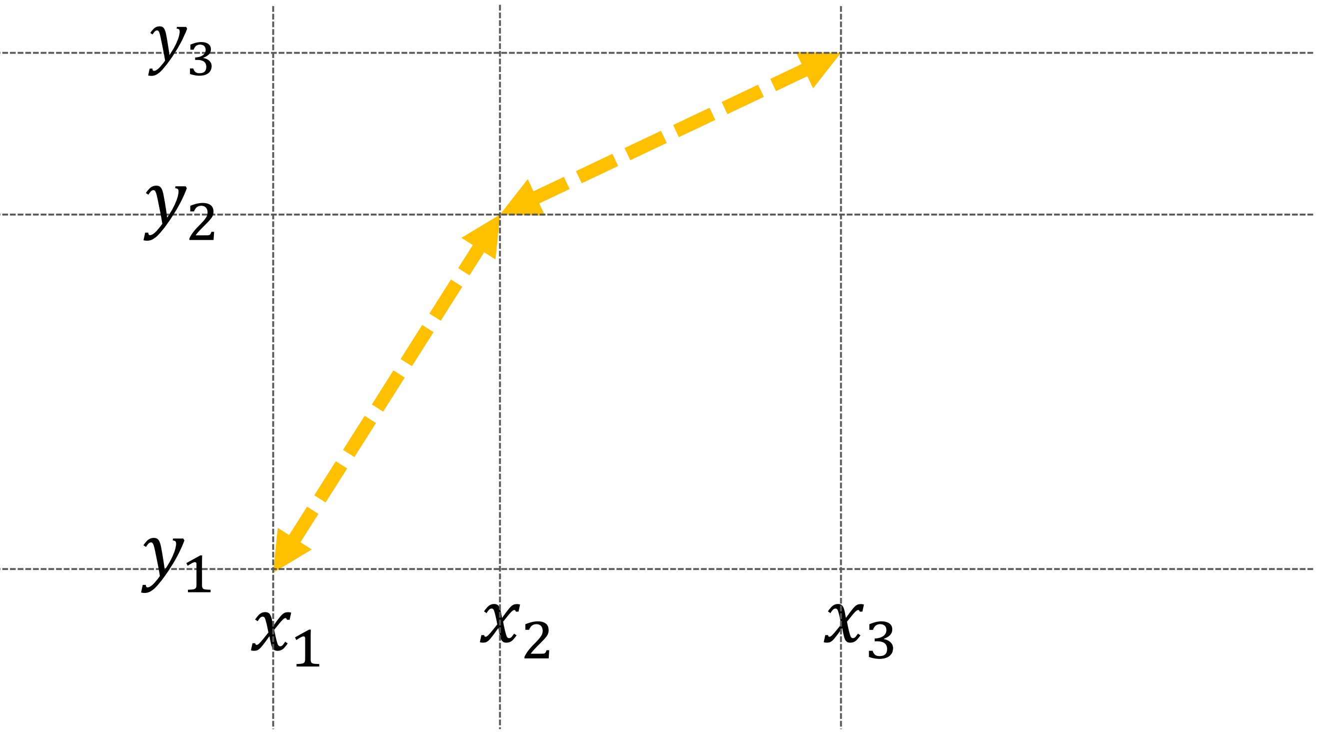
0,0,0,-4,0,0,-2,1,-2,1,0,-2

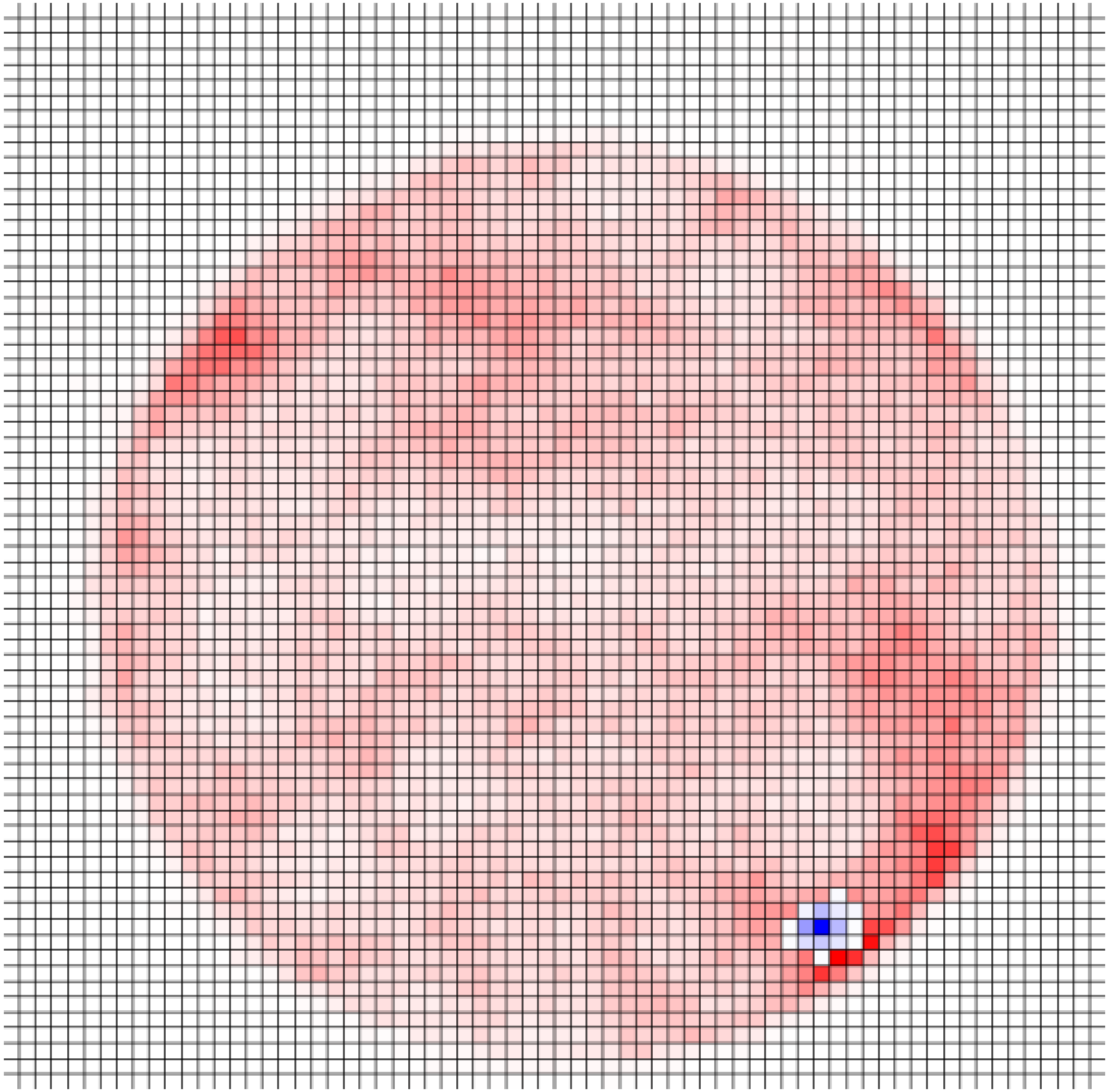
1 1 1 010100 1 1 0011 011 1 0011 1 011

$$v_1 = (x_2 - x_1, y_2 - y_1)$$

$$v_2 = (x_3 - x_2, y_3 - y_2)$$

$$a = (2x_2 - x_1 - x_3, 2y_2 - y_1 - y_3)$$





~~$\dot{x}, \dot{y}$~~

~~x, y~~

Acceleration Histogram