Instituto Politécnico Nacional,
Escuela Superior de Cómputo

Autómata Celulares

Sistemas Complejos

Oscar Andrés Rosas Hernandez

4 de marzo de 2020

Índice general

Parte I

Marco Teórico

Capítulo 1

Autómata celulares

1.1. Definición

Un autómata celular es un sistema dinámico discreto que consiste en una red regular de autómatas (celdas) de estado finito que cambian sus estados dependiendo de los estados de sus vecinos (y del mismo), de acuerdo con una función de transferencia.

Todas las células cambian su estado simultáneamente usando la misma regla de actualización. El proceso se repite en pasos de tiempo discretos. Resulta que con reglas de actualización sorprendentemente simples se pueden producir dinámicas extremadamente complejas como en el famoso Juego de la vida de John Conway. [1]

El aspecto que mas los caracteriza es su capacidad de lograr una serie de propiedades que surgen de la propia dinámica local a través del paso del tiempo y no desde un inicio, aplicándose a todo el sistema en general. Por lo tanto, no es fácil analizar las propiedades globales de un AC desde su comienzo, complejo por naturaleza, si no es por medio de una simulación, partiendo de un estado o configuración inicial de células y cambiando en cada instante los estados de todas ellas de forma síncrona.

1.1.1. Caracteristicas

- Son discretos tanto en tiempo como en espacio
- Son homogeneos tanto en tiempo como en espacio (la misma regla es aplicada a todas las celulas al mismo tiempo)
- Sus interacciones son locales

Para especificar una autómata celular, debemos especificar los siguientes elementos (algunos de los cuales pueden ser claros por el contexto):

- La dimensión $d \in Z^+$,
- El conjunto de estados finitos S
- Una vecindad N de celdas
- La función de activación $f : S^m \to S$

Por lo tanto, definimos formalmente a un autómata celular correspondiente como la 4-tupla $A = (d, S, N, f)$.

1.2. Autómata celulares elementales

Los autómatas elementales son autómatas celulares unidimensionales con dos estados y una vecindad de radio 1: $d = 1$, $S = \{ 0, 1 \}$, $N = (-1, 0, 1)$.

Se diferencian entre sí solo en la elección de la regla f. Hay 256 autómata elementales.

1.3. Clases de Wolfram

S.Wolfram trabajó en los años 80 con los autómatas elementales y basándose en observaciones empíricas de su comportamiento en configuraciones iniciales aleatorias, las clasificó en cuatro clases.

Estas se conocen como clases Wolfram. Las definiciones no son matemáticamente rigurosas, y desde entonces se han propuesto clasificaciones más precisas.

Wolfram definió las clases de la siguiente manera:

- (W1): Casi todas las configuraciones iniciales conducen a la misma configuración uniforme de punto fijo o a un estado homogeneo. Casi todos los patrones iniciales evolucionan rápidamente en un estado estable y homogéneo. Cualquier aleatoriedad en el patrón inicial desaparece.

  Son aquellos cuya evolución eventualmente conduce a células de un solo tipo.

- (W2): Casi todas las configuraciones iniciales conducen a una configuración que se repite periódicamente o estructuras simples. Casi todos los patrones iniciales evolucionan rápidamente hacia estructuras estables u oscilantes. Parte de la aleatoriedad del patrón inicial puede permanecer, pero solo algunos restos. Los cambios locales en el patrón inicial tienden a permanecer locales.

- (W3): Casi todas las configuraciones iniciales conducen a un comportamiento esencialmente aleatorio o caoticos Casi todos los patrones iniciales evolucionan de forma pseudo-aleatoria o caótica. Las estructuras estables que aparecen son destruidas rápidamente por el ruido circundante. Los cambios locales en el patrón inicial tienden a propagarse indefinidamente.

- (W4): Surgen estructuras localizadas con interacciones complejas y localizadas veces de larga vida. Casi todos los patrones iniciales evolucionan en las estructuras que interactúan de manera compleja e interesante, con la formación de las estructuras locales que son capaces de sobrevivir por largos períodos de tiempo.

  Podría ser el caso de que apareciesen estructuras estables u oscilantes, pero el número de pasos necesarios para llegar a este estado puede ser muy grande, incluso cuando el patrón inicial es relativamente simple.

Los cambios locales en el patrón inicial pueden extenderse indefinidamente.

Puede exhibir cualquiera de los comportamientos anteriores simultáneamente y parecen poseer el tipo de complejidad que se encuentra en la intersección entre modelado matemático y estudios de vida.

[1]

Es importante recalcar que Wolfram las definia de tal manera que la regla 2 estuviera contenida en al regla 1 y asi con la regla 3 y la 2 y la 4 y la 3.

Wolfram conjeturó que los autómatas celulares de esa clase (W4) son computacionalmente universales. (cosa que se ha probado para la regla 110).

Algo interesante es que leyendo sobre posibles clasificaciones descubrí que en 1988 un grupo de investigadores descubrio redefiniendo formalmente las clases de Wolfram resulta ser indecidible determinar a qué clase pertenece un automata. [4]

Capítulo 2

La reducción de 256 reglas a solo 88

En su libro A new Kind of Science, S.Wolfram [2] logra demostrar de manera bastante trivial que las 356 reglas se pueden reducir a solo 88 reglas irreducibles, para hacerlo se basa en una idea bastante sencilla:

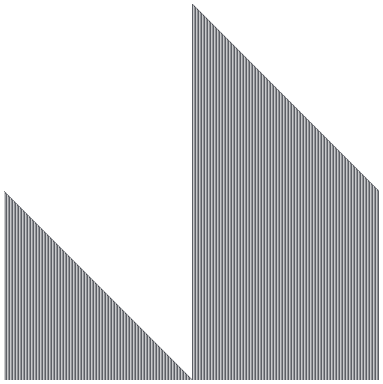
Tienes una regla n entonces:

- La regla que cambia todos los unos por cero no cambia la naturaleza de la regla
- La regla que cambia derecha por izquierda no cambia la naturaleza de la regla
- La regla que cambia las dos anteriores al mismo tiempo no cambia la naturaleza de la regla

Asi podemos hacer grupos de 4, este programa en C++ nos permite obtener las clases de manera sencilla:

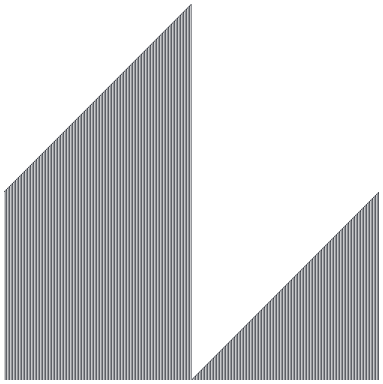
Mira por ejemplo estas 4 clases:

(a) 28



(b) 199

(c) 70



(d) 157

Estas son las clases equivalentes:

- 0, 255, 0, 255
- 1, 127, 1, 127
- 2, 191, 16, 247
- 3, 63, 17, 119
- 4, 223, 4, 223
- 5, 95, 5, 95
- 6, 159, 20, 215
- 7, 31, 21, 87
- 8, 239, 64, 253
- 9, 111, 65, 125
- 10, 175, 80, 245
- 11, 47, 81, 117
- 12, 207, 68, 221
- 13, 79, 69, 93
- 14, 143, 84, 213
- 15, 15, 85, 85
- 16, 247, 2, 191
- 17, 119, 3, 63
- 18, 183, 18, 183
- 19, 55, 19, 55
- 20, 215, 6, 159
- 21, 87, 7, 31
- 22, 151, 22, 151
- 23, 23, 23, 23
- 24, 231, 66, 189
- 25, 103, 67, 61
- 26, 167, 82, 181
- 27, 39, 83, 53
- 28, 199, 70, 157
- 29, 71, 71, 29
- 30, 135, 86, 149
- 31, 7, 87, 21
- 32, 251, 32, 251
- 33, 123, 33, 123
- 34, 187, 48, 243
- 35, 59, 49, 115
- 36, 219, 36, 219
- 37, 91, 37, 91
- 38, 155, 52, 211
- 39, 27, 53, 83
- 40, 235, 96, 249
- 41, 107, 97, 121
- 42, 171, 112, 241
- 43, 43, 113, 113
- 44, 203, 100, 217
- 45, 75, 101, 89
- 46, 139, 116, 209
- 47, 11, 117, 81
- 48, 243, 34, 187
- 49, 115, 35, 59
- 50, 179, 50, 179
- 51, 51, 51, 51
- 52, 211, 38, 155
- 53, 83, 39, 27
- 54, 147, 54, 147
- 55, 19, 55, 19
- 56, 227, 98, 185
- 57, 99, 99, 57
- 58, 163, 114, 177
- 59, 35, 115, 49
- 60, 195, 102, 153
- 61, 67, 103, 25
- 62, 131, 118, 145
- 63, 3, 119, 17
- 64, 253, 8, 239
- 65, 125, 9, 111
- 66, 189, 24, 231
- 67, 61, 25, 103
- 68, 221, 12, 207
- 69, 93, 13, 79
- 70, 157, 28, 199
- 71, 29, 29, 71
- 72, 237, 72, 237
- 73, 109, 73, 109
- 74, 173, 88, 229
- 75, 45, 89, 101

- 76, 205, 76, 205
- 77, 77, 77, 77
- 78, 141, 92, 197
- 79, 13, 93, 69
- 80, 245, 10, 175
- 81, 117, 11, 47
- 82, 181, 26, 167
- 83, 53, 27, 39
- 84, 213, 14, 143
- 85, 85, 15, 15
- 86, 149, 30, 135
- 87, 21, 31, 7
- 88, 229, 74, 173
- 89, 101, 75, 45
- 90, 165, 90, 165
- 91, 37, 91, 37
- 92, 197, 78, 141
- 93, 69, 79, 13
- 94, 133, 94, 133
- 95, 5, 95, 5
- 96, 249, 40, 235
- 97, 121, 41, 107
- 98, 185, 56, 227
- 99, 57, 57, 99
- 100, 217, 44, 203
- 101, 89, 45, 75
- 102, 153, 60, 195
- 103, 25, 61, 67
- 104, 233, 104, 233
- 105, 105, 105, 105
- 106, 169, 120, 225
- 107, 41, 121, 97
- 108, 201, 108, 201
- 109, 73, 109, 73
- 110, 137, 124, 193
- 111, 9, 125, 65
- 112, 241, 42, 171
- 113, 113, 43, 43
- 114, 177, 58, 163
- 115, 49, 59, 35
- 116, 209, 46, 139
- 117, 81, 47, 11
- 118, 145, 62, 131
- 119, 17, 63, 3
- 120, 225, 106, 169
- 121, 97, 107, 41
- 122, 161, 122, 161
- 123, 33, 123, 33
- 124, 193, 110, 137
- 125, 65, 111, 9
- 126, 129, 126, 129
- 127, 1, 127, 1
- 128, 254, 128, 254
- 129, 126, 129, 126
- 130, 190, 144, 246
- 131, 62, 145, 118
- 132, 222, 132, 222
- 133, 94, 133, 94
- 134, 158, 148, 214
- 135, 30, 149, 86
- 136, 238, 192, 252
- 137, 110, 193, 124
- 138, 174, 208, 244
- 139, 46, 209, 116
- 140, 206, 196, 220
- 141, 78, 197, 92
- 142, 142, 212, 212
- 143, 14, 213, 84
- 144, 246, 130, 190
- 145, 118, 131, 62
- 146, 182, 146, 182
- 147, 54, 147, 54
- 148, 214, 134, 158
- 149, 86, 135, 30
- 150, 150, 150, 150
- 151, 22, 151, 22
- 152, 230, 194, 188
- 153, 102, 195, 60
- 154, 166, 210, 180

- 155, 38, 211, 52
- 156, 198, 198, 156
- 157, 70, 199, 28
- 158, 134, 214, 148
- 159, 6, 215, 20
- 160, 250, 160, 250
- 161, 122, 161, 122
- 162, 186, 176, 242
- 163, 58, 177, 114
- 164, 218, 164, 218
- 165, 90, 165, 90
- 166, 154, 180, 210
- 167, 26, 181, 82
- 168, 234, 224, 248
- 169, 106, 225, 120
- 170, 170, 240, 240
- 171, 42, 241, 112
- 172, 202, 228, 216
- 173, 74, 229, 88
- 174, 138, 244, 208
- 175, 10, 245, 80
- 176, 242, 162, 186
- 177, 114, 163, 58
- 178, 178, 178, 178
- 179, 50, 179, 50
- 180, 210, 166, 154
- 181, 82, 167, 26
- 182, 146, 182, 146
- 183, 18, 183, 18
- 184, 226, 226, 184
- 185, 98, 227, 56
- 186, 162, 242, 176
- 187, 34, 243, 48
- 188, 194, 230, 152
- 189, 66, 231, 24
- 190, 130, 246, 144
- 191, 2, 247, 16
- 192, 252, 136, 238
- 193, 124, 137, 110
- 194, 188, 152, 230
- 195, 60, 153, 102
- 196, 220, 140, 206
- 197, 92, 141, 78
- 198, 156, 156, 198
- 199, 28, 157, 70
- 200, 236, 200, 236
- 201, 108, 201, 108
- 202, 172, 216, 228
- 203, 44, 217, 100
- 204, 204, 204, 204
- 205, 76, 205, 76
- 206, 140, 220, 196
- 207, 12, 221, 68
- 208, 244, 138, 174
- 209, 116, 139, 46
- 210, 180, 154, 166
- 211, 52, 155, 38
- 212, 212, 142, 142
- 213, 84, 143, 14
- 214, 148, 158, 134
- 215, 20, 159, 6
- 216, 228, 202, 172
- 217, 100, 203, 44
- 218, 164, 218, 164
- 219, 36, 219, 36
- 220, 196, 206, 140
- 221, 68, 207, 12
- 222, 132, 222, 132
- 223, 4, 223, 4
- 224, 248, 168, 234
- 225, 120, 169, 106
- 226, 184, 184, 226
- 227, 56, 185, 98
- 228, 216, 172, 202
- 229, 88, 173, 74
- 230, 152, 188, 194
- 231, 24, 189, 66
- 232, 232, 232, 232
- 233, 104, 233, 104

- 234, 168, 248, 224
- 235, 40, 249, 96
- 236, 200, 236, 200
- 237, 72, 237, 72
- 238, 136, 252, 192
- 239, 8, 253, 64
- 240, 240, 170, 170
- 241, 112, 171, 42
- 242, 176, 186, 162
- 243, 48, 187, 34
- 244, 208, 174, 138
- 245, 80, 175, 10
- 246, 144, 190, 130
- 247, 16, 191, 2
- 248, 224, 234, 168
- 249, 96, 235, 40
- 250, 160, 250, 160
- 251, 32, 251, 32
- 252, 192, 238, 136
- 253, 64, 239, 8
- 254, 128, 254, 128
- 255, 0, 255, 0

Eligiendo la regla mas pequeña de cada clase podemos general el conjunto de reglas que vamos a analizar: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 18, 19, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 35, 36, 37, 38, 40, 41, 42, 43, 44, 45, 46, 50, 51, 54, 56, 57, 58, 60, 62, 72, 73, 74, 76, 77, 78, 90, 94, 104, 105, 106, 108, 110, 122, 126, 128, 130, 132, 134, 136, 138, 140, 142, 146, 150, 152, 154, 156, 160, 162, 164, 168, 170, 172, 178, 184, 200, 204, 232}.

El código realizado fue:

```
#include <algorithm>
#include <bitset>
#include <cstdint>
#include <iostream>
#include <set>

using namespace std;
using num = uint8_t;

auto flip_bits(const num n) -> num {
  auto bits = bitset<8>{n};
  auto buffer = bits;

  buffer[0] = not bits[7];
  buffer[1] = not bits[6];
  buffer[2] = not bits[5];
  buffer[3] = not bits[4];
  buffer[4] = not bits[3];
  buffer[5] = not bits[2];
  buffer[6] = not bits[1];
  buffer[7] = not bits[0];

  return buffer.to_ulong();
}

auto flip_dir(const num n) -> num {
  auto bits = bitset<8>{n};
  auto buffer = bits;

  buffer[1] = bits[4];
  buffer[3] = bits[6];
  buffer[6] = bits[3];
  buffer[4] = bits[1];

  return buffer.to_ulong();
}

auto get_copies(const num n) -> tuple<num, num, num> {
  const auto ones = flip_bits(n);
  const auto dirs = flip_dir(n);
  const auto crazy = flip_dir(flip_bits(n));

  return {ones, dirs, crazy};
}
```

```
auto add_id(const set<num> &seen, set<int> &ids, num a, num b, num c, num d) {
  auto representant = 256;

  if (not seen.count(a))
    representant = min<int>(representant, a);
  if (not seen.count(b))
    representant = min<int>(representant, b);
  if (not seen.count(c))
    representant = min<int>(representant, c);
  if (not seen.count(d))
    representant = min<int>(representant, d);

  if (representant != 256)
    ids.insert(representant);
}

auto main() -> int {
  auto seen = set<num>{};
  auto representants = set<int>{};

  auto i = 0;
  for (auto i = 0; i < 256; ++i) {
    num a = i;
    auto [b, c, d] = get_copies(a);

    add_id(seen, representants, a, b, c, d);
    printf("%d %d %d %d\n", a, b, c, d);

    seen.insert({a, b, c, d});
  }

  cout << endl;
  for (auto key : representants) {
    cout << key << endl;
  }

  return 0;
}
```

Compilado con:

```
  g++ -std=c++17 Equivalence.cpp && ./a.out
```

Es trivial comprobar que el codigo funciona comparando con los resultados obtenido por Wolfram [2].

Capítulo 3

Mi clasificación de los autómatas celulares

3.1. Codigo

Para esto ocupamos dos versiones una creada en Typescript para crear un simulador web bastante bonito:

```
type bits = Uint8Array;

class CellularAutomata {
  readonly histogram: Array<number>;
  epoch: number;
  data: bits;
  buffer: bits;

  constructor(init: Array<number>) {
    this.data = new Uint8Array(new ArrayBuffer(init.length));
    this.buffer = new Uint8Array(new ArrayBuffer(init.length));
    this.histogram = [];
    this.epoch = 0;

    let ones = 0;
    for (let i = 0; i < this.data.length; ++i) {
      this.data[i] = init[i];
      if (this.data[i]) ++ones;
    }

    this.histogram[this.epoch] = ones;
  }

  newEpoch(rulesID: number): void {
    this.epoch += 1;
    let ones = 0;

    const rule = this.getRules(rulesID);
    for (let i = 0; i < this.buffer.length; ++i) {
      this.buffer[i] = rule(this.data, i);
      if (this.buffer[i]) ++ones;
    }

    this.histogram[this.epoch] = ones;
    [this.data, this.buffer] = [this.buffer, this.data];
  }

  createNewEpoch(rulesID: number): bits {
    const rule = this.getRules(rulesID);
    return this.data.map((_, i) => rule(this.data, i));
  }

  getRules = (rulesID: number) => (data: bits, index: number): number => {
    const limit = data.length - 1;
    const n1 = index === 0 ? limit : index - 1;
```

```
    const n2 = index === limit ? 0 : index + 1;

    const id = (data[n1] << 2) + (data[index] << 1) + (data[n2] << 0);
    return (rulesID >> id) & 1;
  };

  get average(): number {
    let total = 0;
    for (let i = 0; i < this.histogram.length; ++i) total += this.histogram[i];

    return total / this.histogram.length;
  }

  get variance(): number {
    const average = this.average;
    let variance = 0;

    for (let i = 0; i < this.histogram.length; ++i)
      variance += (this.histogram[i] - average) * (this.histogram[i] - average);

    return variance / this.histogram.length;
  }
}

export default CellularAutomata;
```

Y otra creada en Python para poder tomar velocidad a la hora de crear las gráficas:

```
from PIL import Image
import os
import random
import plotly.io as pio
from math import sqrt

n = iterations = 500


def graph(histogram, average, variance, path):
    global n

    desviation = sqrt(variance)

    trace1 = {"type": "bar", "y": histogram, "opacity": 1,
              "name": "Histogram", "marker": {"color": "rgb(158,202,225)"}, }

    trace2 = {"type": "scatter", "y": [average, average], "x": [0, len(histogram) - 1], "opacity": 0.5,
              "name": "Average", "marker": {"color": "#FC7A7A"}, }

    trace3 = {"type": "scatter", "y": [average + desviation, average + desviation], "x": [0, len(histogram) - 1],
     "opacity": 0.5,
              "name": "sqrt derivation +", "marker": {"color": "#009999"}, }

    trace4 = {"type": "scatter", "y": [average - desviation, average - desviation], "x": [0, len(histogram) - 1],
     "opacity": 0.5,
              "name": "sqrt derivation -", "marker": {"color": "#009999"}, }

    fig = { "data": [trace1, trace2, trace3, trace4],  "layout": {"title": {"text": "Ones"}} }

    pio.write_image(fig, path)


def get_rules(rules_id):
    def rule(s, i):
        limit = len(s) - 1
        n1 = limit if i == 0 else i - 1
        n2 = 0 if i == limit else i + 1

        id = (s[n1] << 2) + (s[i] << 1) + (s[n2] << 0)
        return (rules_id >> id) & 1

    return rule


def print_evolution(steps, s, rules_id, pixels, path):
    global n
    histogram = []

    rules = get_rules(rules_id)
    current, temporal = list(s), list(s)
    limit = len(s)

    one, zero = (255, 255, 255), (40, 44, 52)

    for step in range(steps):
        histogram.append(sum(current))
```

```
        for i in range(limit):
            temporal[i] = rules(current, i)
        for j, val in enumerate(temporal):
            pixels[j, step] = one if val else zero

        current = list(temporal)

    average = sum(histogram) / n

    variance = 0
    for val in histogram:
        variance += (val - average) * (val - average)

    variance = variance / n

    graph(histogram, average, variance, path)


rules = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 18, 19, 22, 23, 24, 25, 26, 27, 28, 29,
         30, 32, 33, 34, 35, 36, 37, 38, 40, 41, 42, 43, 44, 45, 46, 50, 51, 54, 56, 57, 58, 60, 62, 72, 73,
         74, 76, 77, 78, 90, 94, 104, 105, 106, 108, 110, 122, 126, 128, 130, 132, 134, 136, 138, 140, 142,
         146, 150, 152, 154, 156, 160, 162, 164, 168, 170, 172, 178, 184, 200, 204, 232]

inits = []
random.seed(10)


def oneMiddle():
    s = [0] * n
    s[n // 2] = 1
    return s


def getPercentage(percentage):
    s = [0] * n
    for i in range(len(s)):
        s[i] = 1 if percentage > random.random() else 0

    return s


inits = [oneMiddle(), getPercentage(0.08), getPercentage(.5), getPercentage(.87)]

for rule in [28, 199, 70, 157]:
    print(rule)
    for i, init in enumerate(inits):
        letter = chr(ord("a") + i)
        img = Image.new('RGB', (500, 500), "black")
        pixels = img.load()


        diagram = f"../Images/{rule}/{letter}.png"
        path = f"../Images/{rule}/dia-{letter}.png"

        print_evolution(iterations, init, rule, pixels, path)
        img.save(diagram)
```

3.2. W1

Casi todas las configuraciones iniciales conducen a la misma configuración uniforme de punto fijo o a un estado homogeneo. Casi todos los patrones iniciales evolucionan rápidamente en un estado estable y homogéneo. Cualquier aleatoriedad en el patrón inicial desaparece.

Son aquellos cuya evolución eventualmente conduce a células de un solo tipo.

Estan son muy sencillas de clasificar viendo de los histogramas basicamente caen a 0 o a 1.

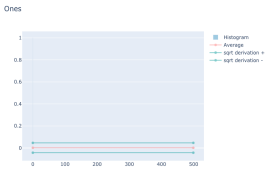
3.2.1. 0

(a) One

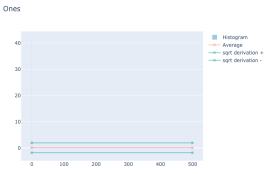
(b) 8 %

(c) 50 %

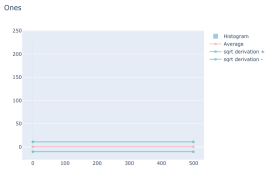
(d) 87 %



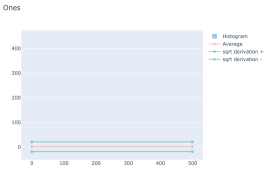
(a) One



(b) 8 %



(c) 50 %



(d) 87 %

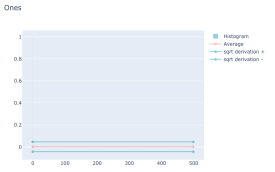
3.2.2. 8

(a) One

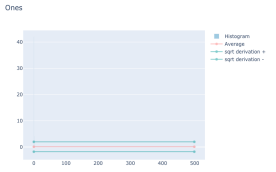
(b) 8 %

(c) 50 %

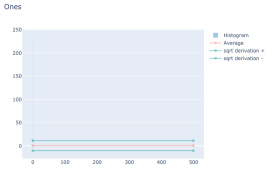
(d) 87 %



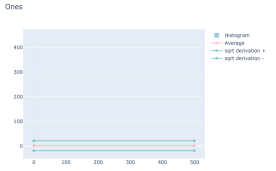
(a) One



(b) 8 %



(c) 50 %



(d) 87 %

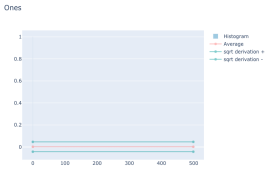
3.2.3. 32

(a) One

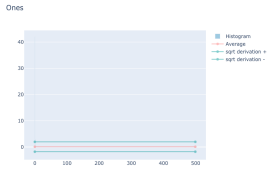
(b) 8 %

(c) 50 %

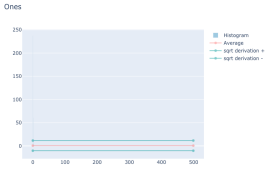
(d) 87 %



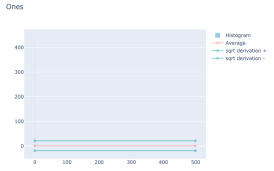
(a) One



(b) 8 %



(c) 50 %



(d) 87 %

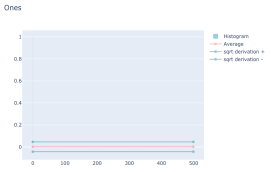
3.2.4. 40

(a) One

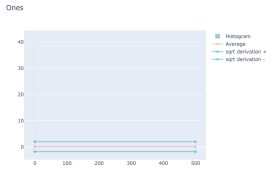
(b) 8 %

(c) 50 %

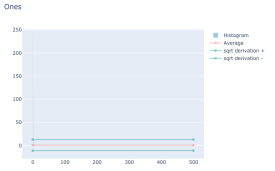
(d) 87 %



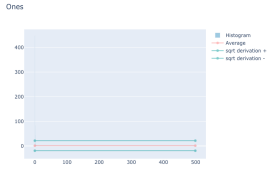
(a) One



(b) 8 %



(c) 50 %



(d) 87 %

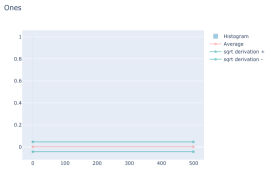
3.2.5. 128

(a) One

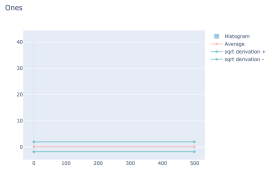
(b) 8 %

(c) 50 %

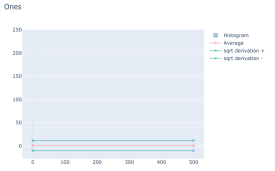
(d) 87 %



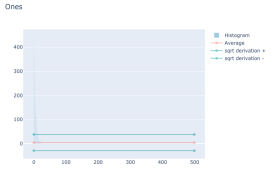
(a) One



(b) 8 %



(c) 50 %



(d) 87 %

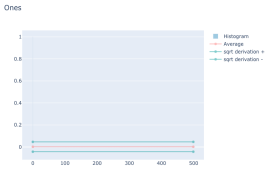
3.2.6. 136

(a) One

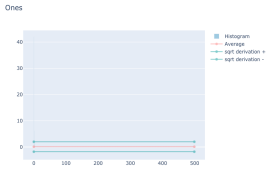
(b) 8 %

(c) 50 %

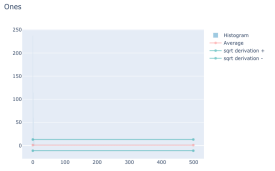
(d) 87 %



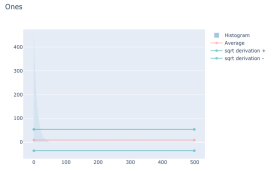
(a) One



(b) 8 %



(c) 50 %



(d) 87 %

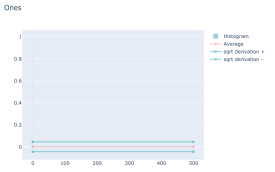
3.2.7. 160

(a) One

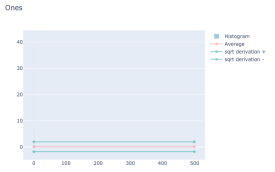
(b) 8 %

(c) 50 %

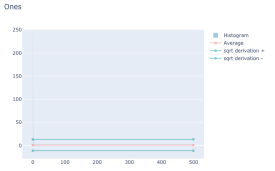
(d) 87 %



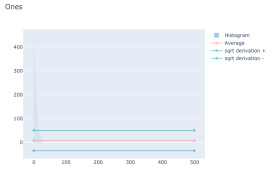
(a) One



(b) 8 %



(c) 50 %



(d) 87 %

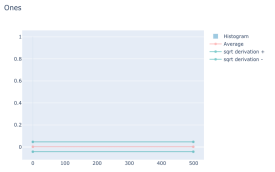
3.2.8. 168

(a) One

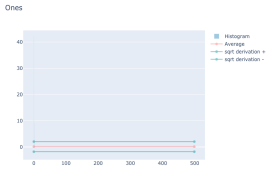
(b) 8 %

(c) 50 %

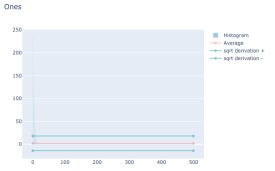
(d) 87 %



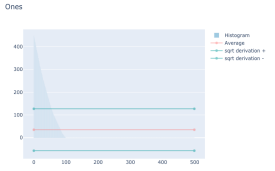
(a) One



(b) 8 %



(c) 50 %



(d) 87 %

3.3. W2

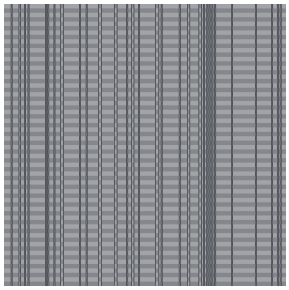
Casi todas las configuraciones iniciales conducen a una configuración que se repite periódicamente o estructuras simples. Casi todos los patrones iniciales evolucionan rápidamente hacia estructuras estables u oscilantes. Parte de la aleatoriedad del patrón inicial puede permanecer, pero solo algunos restos. Los cambios locales en el patrón inicial tienden a permanecer locales.

Estan son muy sencillas de clasificar viendo de los histogramas basicamente caen a 0 o a 1.

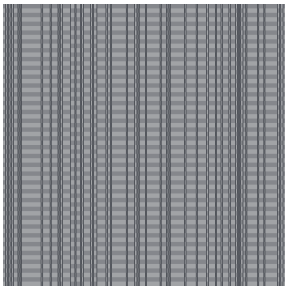
Estos son relativamente sencillos de clasificar basados en el histograma, buscamos aquellos que se estabilizen y la cantidad de 0 sean mas o menos constante.

3.3.1. 1

(a) One

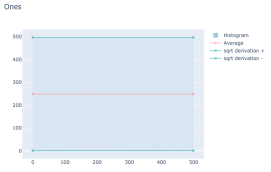


(b) 8 %

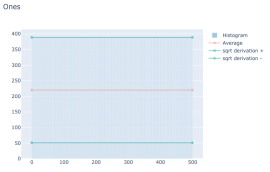


(c) 50 %

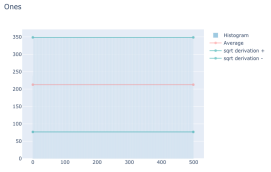
(d) 87 %



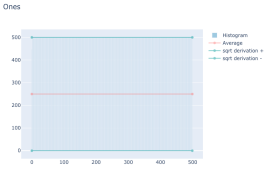
(a) One



(b) 8 %

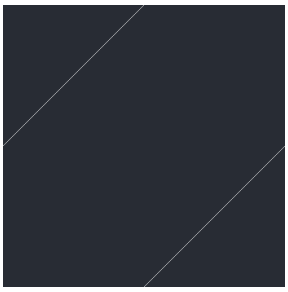


(c) 50 %



(d) 87 %

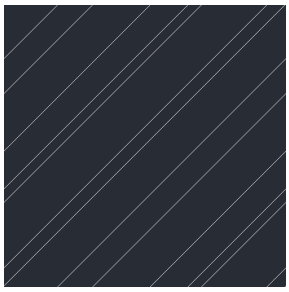
3.3.2. 2



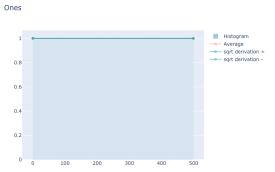
(a) One

(b) 8 %

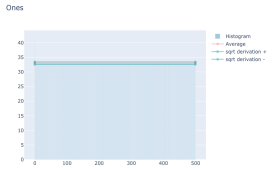
(c) 50 %



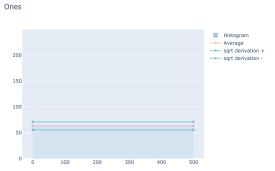
(d) 87 %



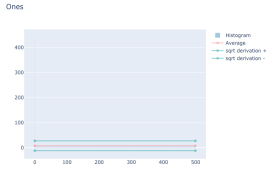
(a) One



(b) 8 %

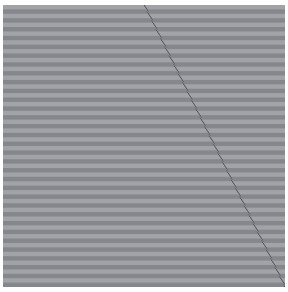


(c) 50 %

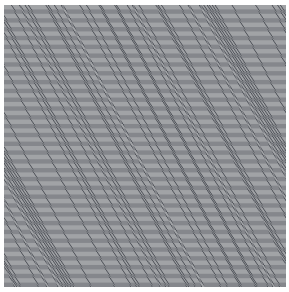


(d) 87 %

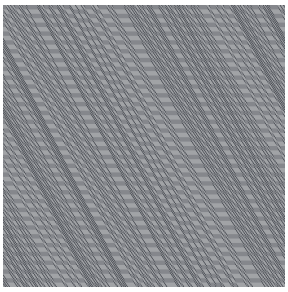
3.3.3. 3



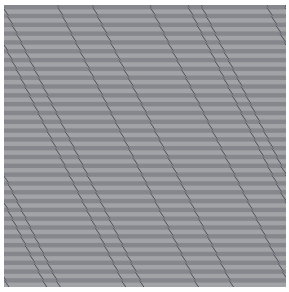
(a) One



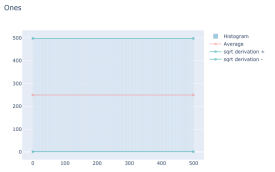
(b) 8 %



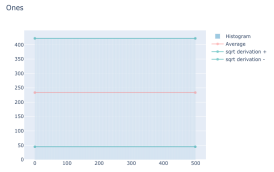
(c) 50 %



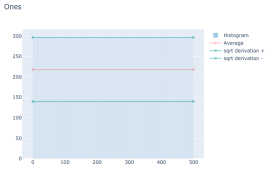
(d) 87 %



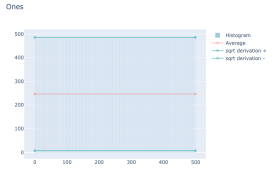
(a) One



(b) 8 %

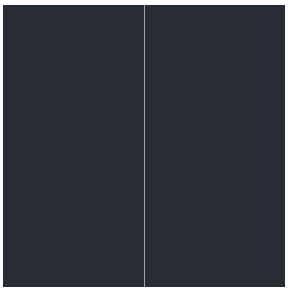


(c) 50 %

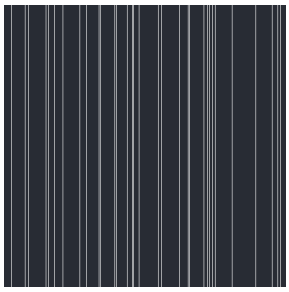


(d) 87 %

3.3.4. 4

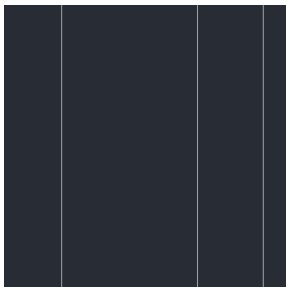


(a) One

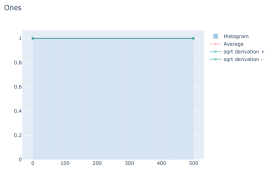


(b) 8 %

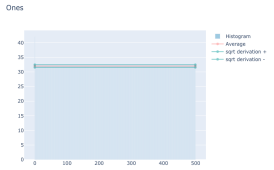
(c) 50 %



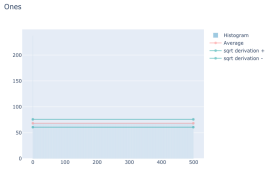
(d) 87 %



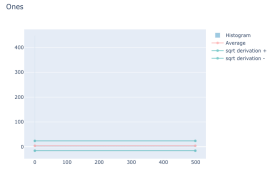
(a) One



(b) 8 %

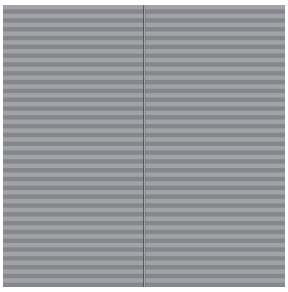


(c) 50 %

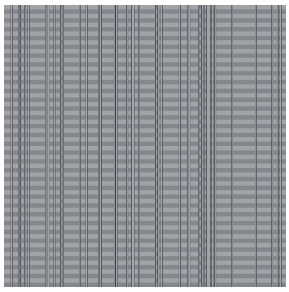


(d) 87 %

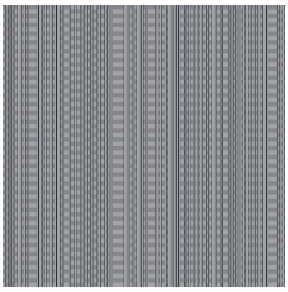
3.3.5. 5



(a) One

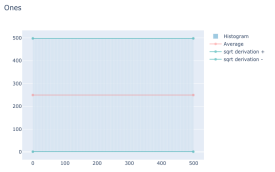


(b) 8 %

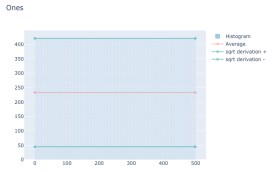


(c) 50 %

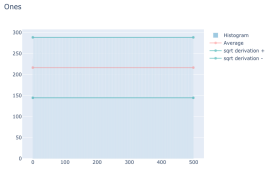
(d) 87 %



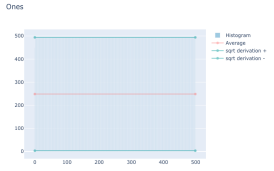
(a) One



(b) 8 %

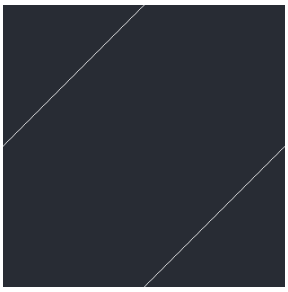


(c) 50 %



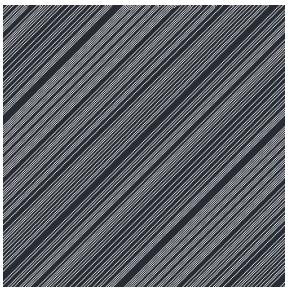
(d) 87 %

3.3.6. 6



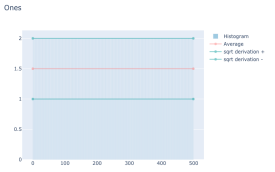
(a) One

(b) 8 %

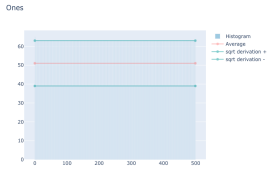


(c) 50 %

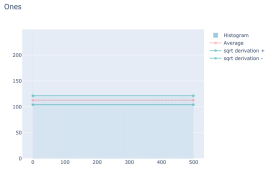
(d) 87 %



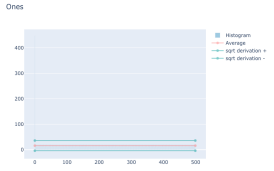
(a) One



(b) 8 %



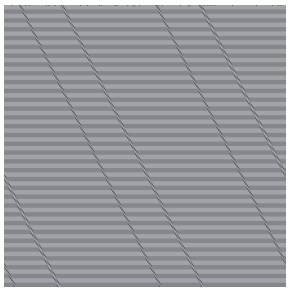
(c) 50 %



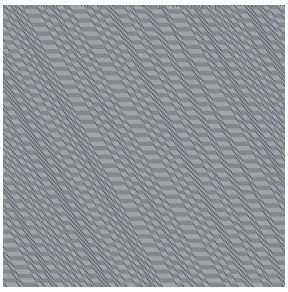
(d) 87 %

3.3.7. 7

(a) One

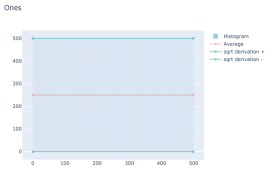


(b) 8 %

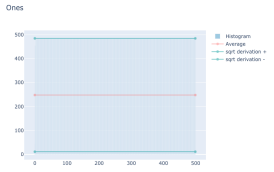


(c) 50 %

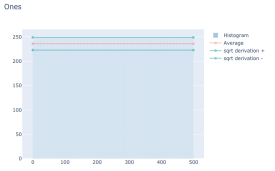
(d) 87 %



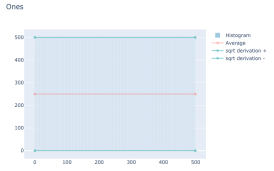
(a) One



(b) 8 %

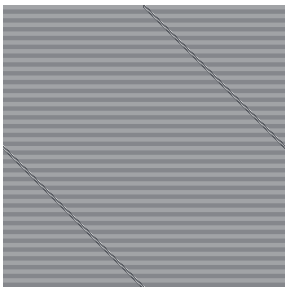


(c) 50 %

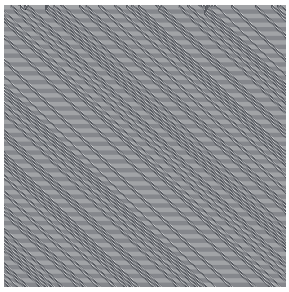


(d) 87 %

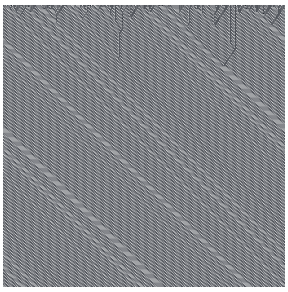
3.3.8. 9



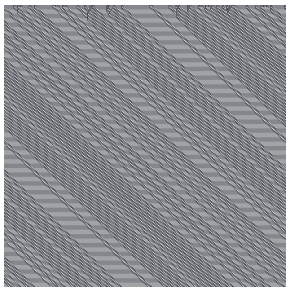
(a) One



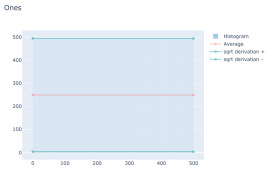
(b) 8 %



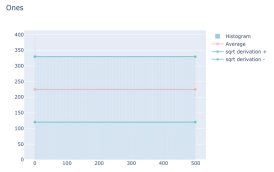
(c) 50 %



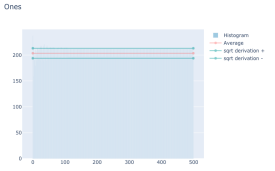
(d) 87 %



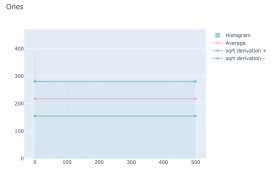
(a) One



(b) 8 %

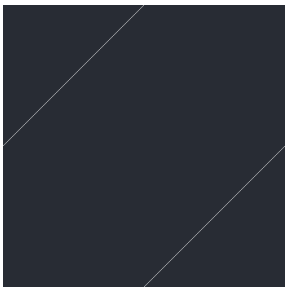


(c) 50 %

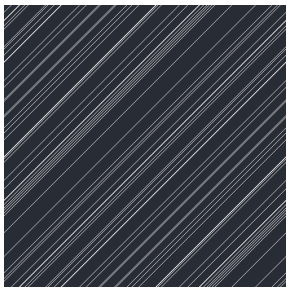


(d) 87 %

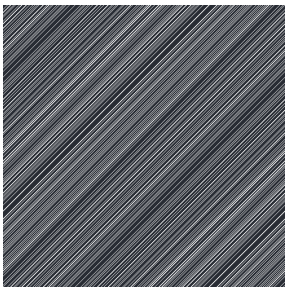
3.3.9. 10



(a) One

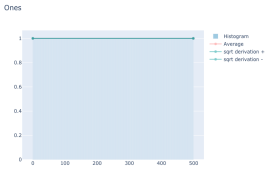


(b) 8 %

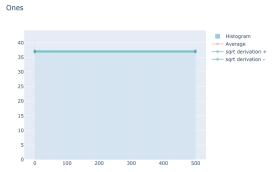


(c) 50 %

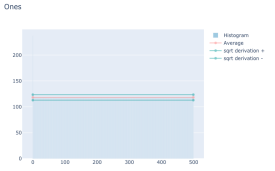
(d) 87 %



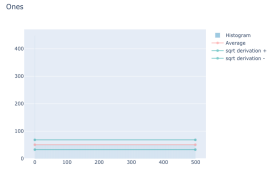
(a) One



(b) 8 %

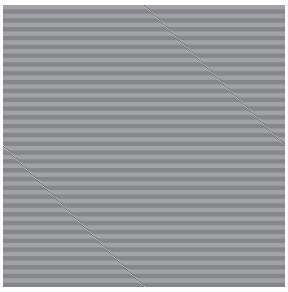


(c) 50 %

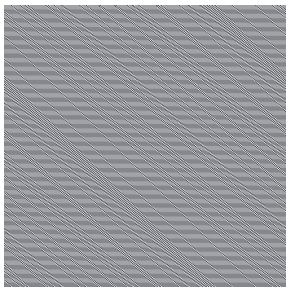


(d) 87 %

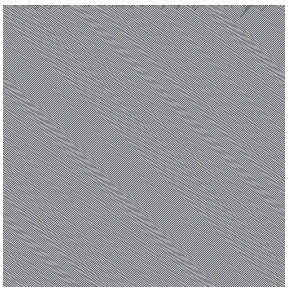
3.3.10. 11



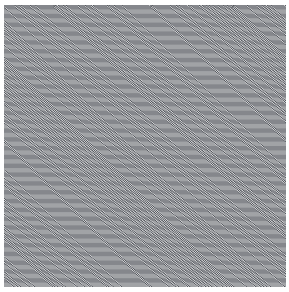
(a) One



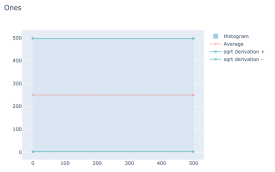
(b) 8 %



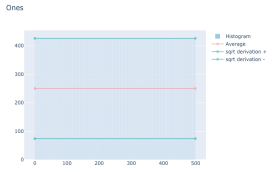
(c) 50 %



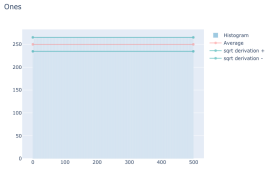
(d) 87 %



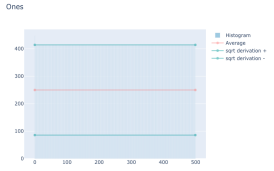
(a) One



(b) 8 %

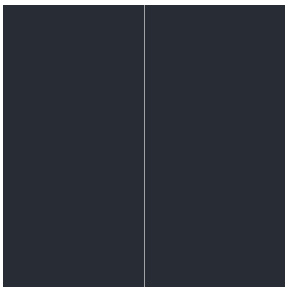


(c) 50 %



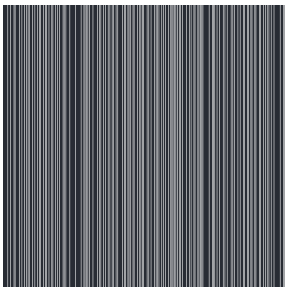
(d) 87 %

3.3.11. 12



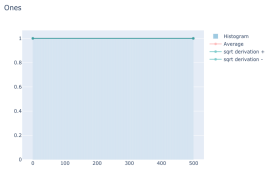
(a) One

(b) 8 %

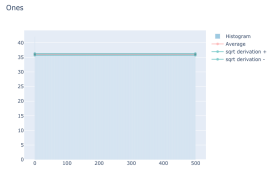


(c) 50 %

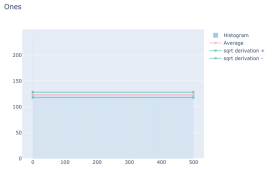
(d) 87 %



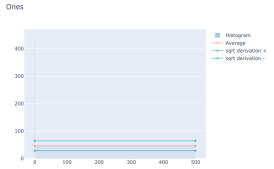
(a) One



(b) 8 %

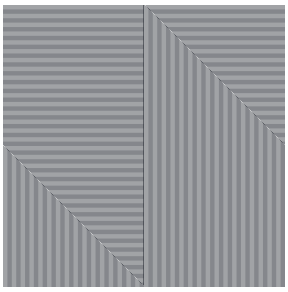


(c) 50 %

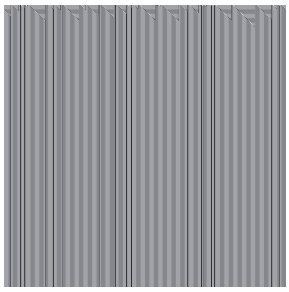


(d) 87 %

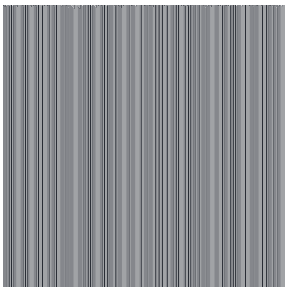
3.3.12. 13



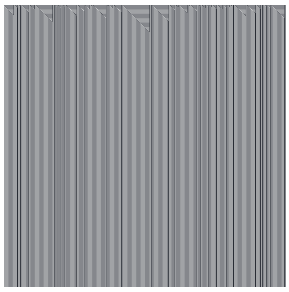
(a) One



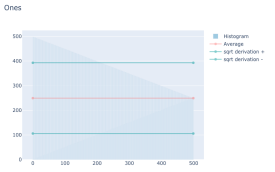
(b) 8 %



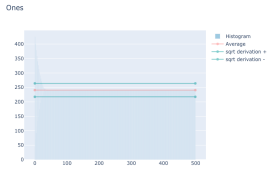
(c) 50 %



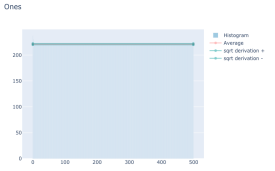
(d) 87 %



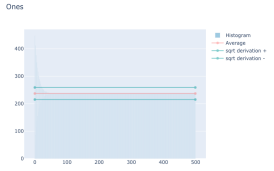
(a) One



(b) 8 %

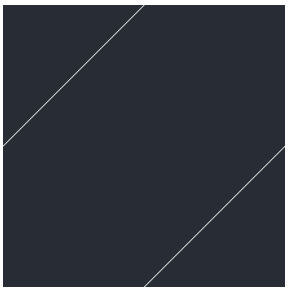


(c) 50 %



(d) 87 %

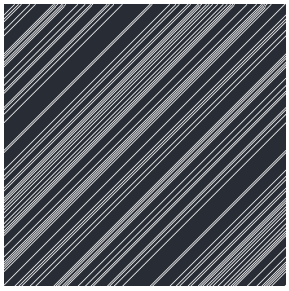
3.3.13. 14



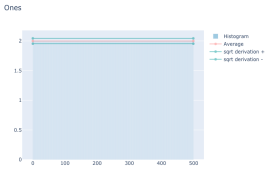
(a) One

(b) 8 %

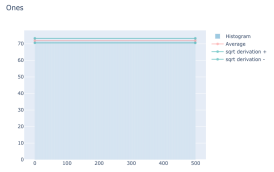
(c) 50 %



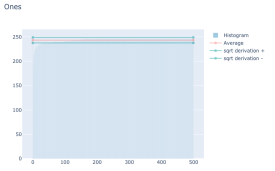
(d) 87 %



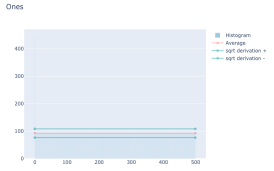
(a) One



(b) 8 %

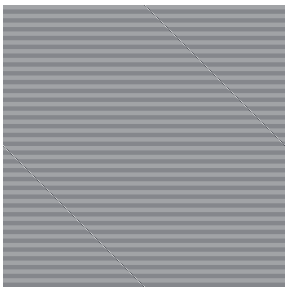


(c) 50 %

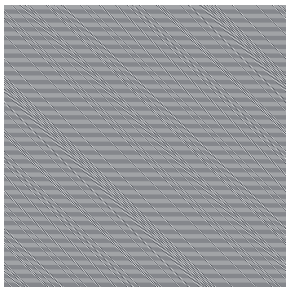


(d) 87 %

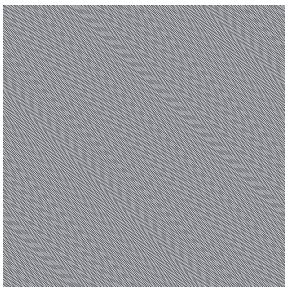
3.3.14. 15



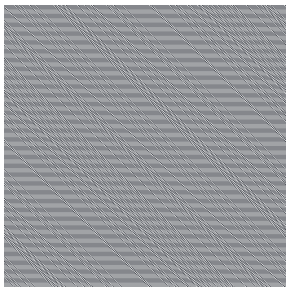
(a) One



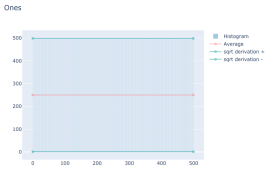
(b) 8 %



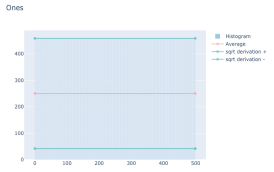
(c) 50 %



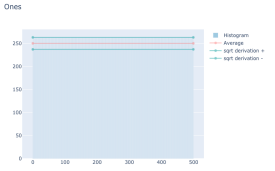
(d) 87 %



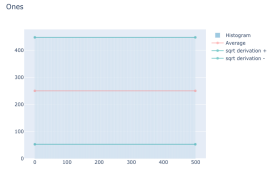
(a) One



(b) 8 %



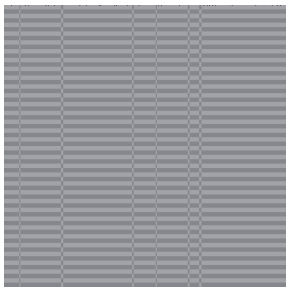
(c) 50 %



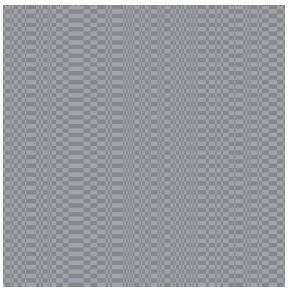
(d) 87 %

3.3.15. 19

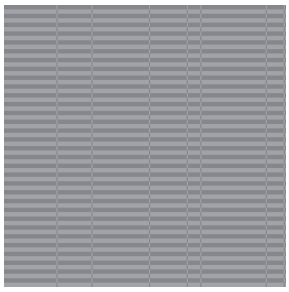
(a) One



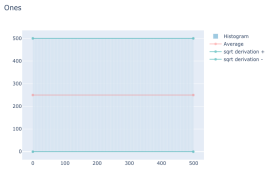
(b) 8 %



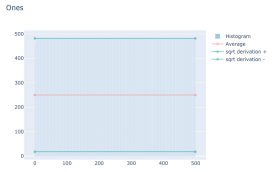
(c) 50 %



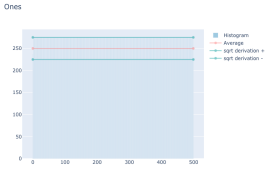
(d) 87 %



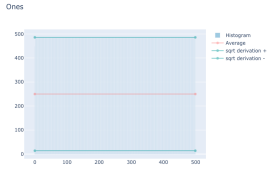
(a) One



(b) 8 %



(c) 50 %

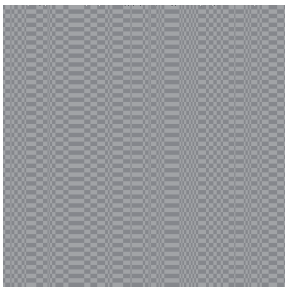


(d) 87 %

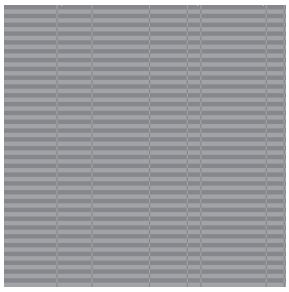
3.3.16. 23

(a) One

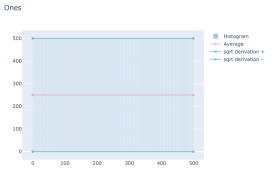
(b) 8 %



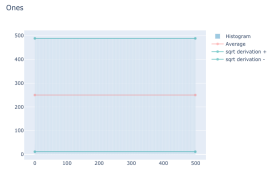
(c) 50 %



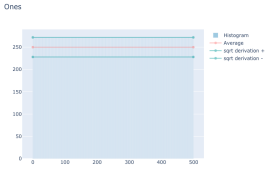
(d) 87 %



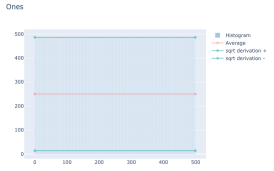
(a) One



(b) 8 %



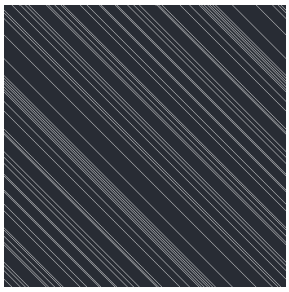
(c) 50 %



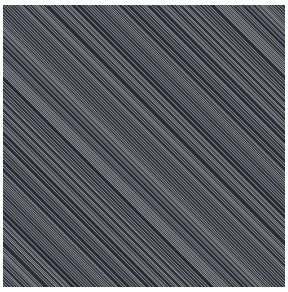
(d) 87 %

3.3.17. 24

(a) One

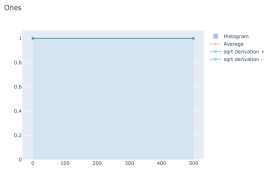


(b) 8 %

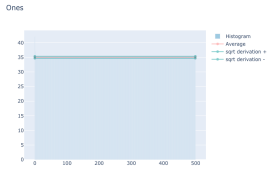


(c) 50 %

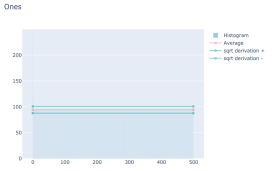
(d) 87 %



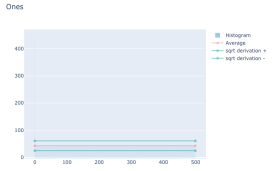
(a) One



(b) 8 %

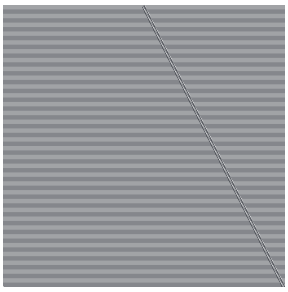


(c) 50 %

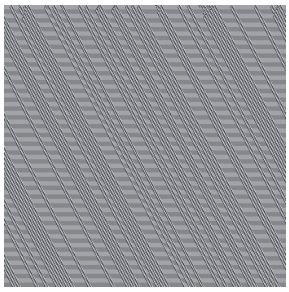


(d) 87 %

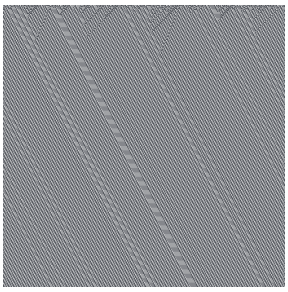
3.3.18. 25



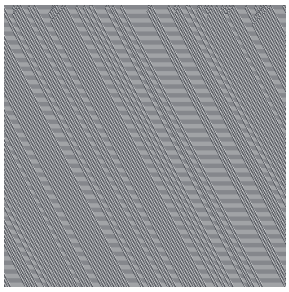
(a) One



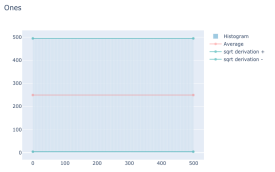
(b) 8 %



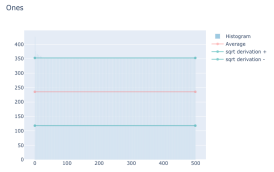
(c) 50 %



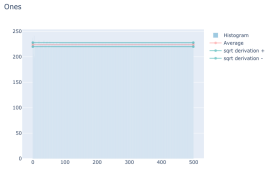
(d) 87 %



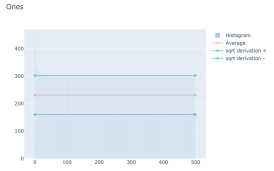
(a) One



(b) 8 %

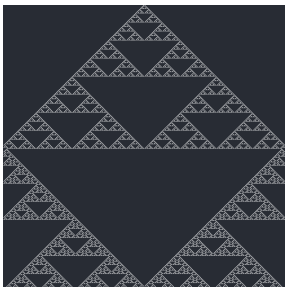


(c) 50 %

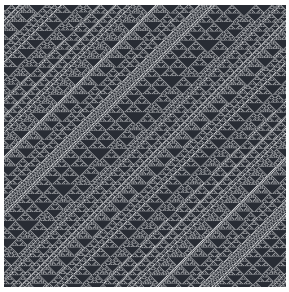


(d) 87 %

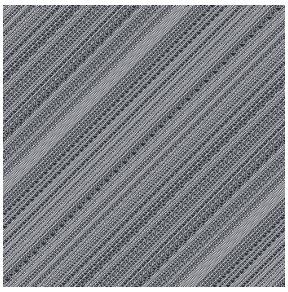
3.3.19. 26



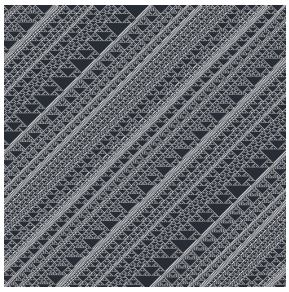
(a) One



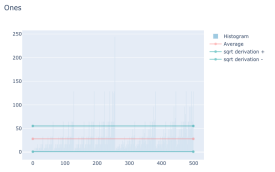
(b) 8 %



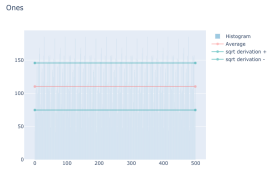
(c) 50 %



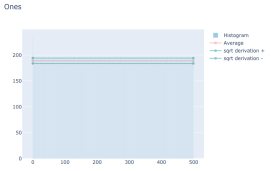
(d) 87 %



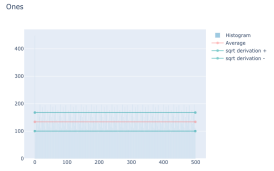
(a) One



(b) 8 %

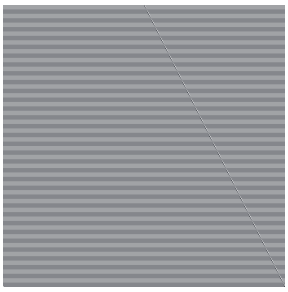


(c) 50 %

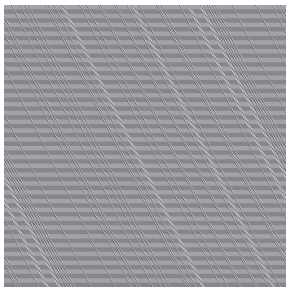


(d) 87 %

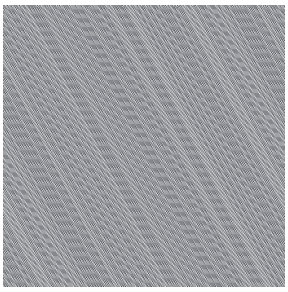
3.3.20. 27



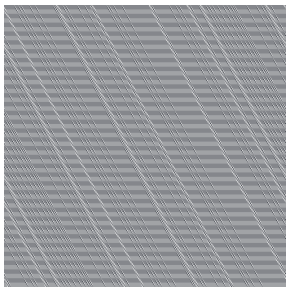
(a) One



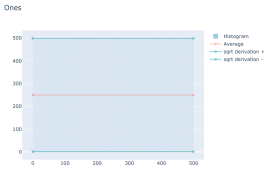
(b) 8 %



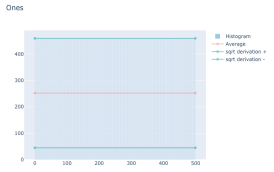
(c) 50 %



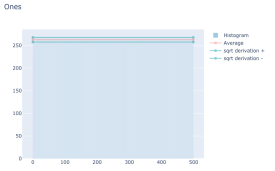
(d) 87 %



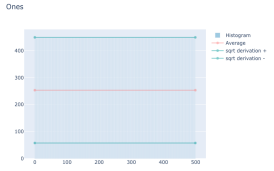
(a) One



(b) 8 %



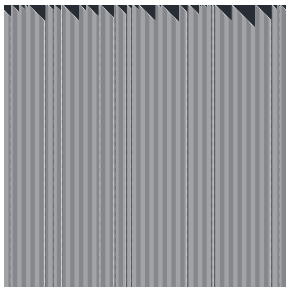
(c) 50 %



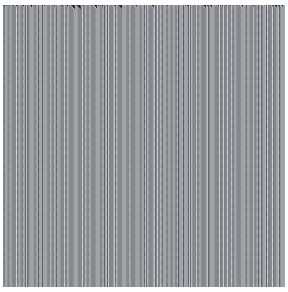
(d) 87 %

3.3.21. 28

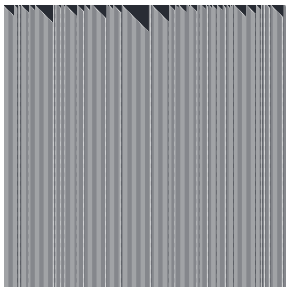
(a) One



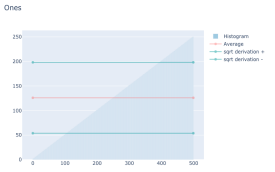
(b) 8 %



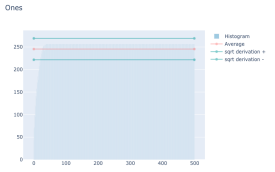
(c) 50 %



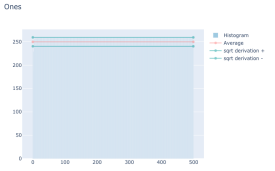
(d) 87 %



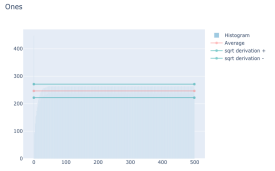
(a) One



(b) 8 %

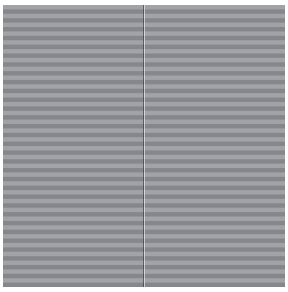


(c) 50 %

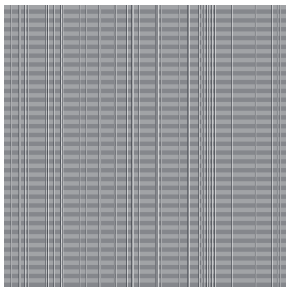


(d) 87 %

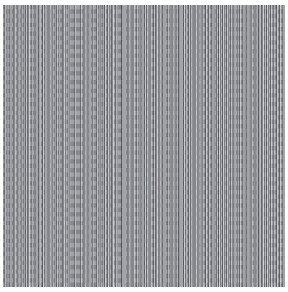
3.3.22. 29



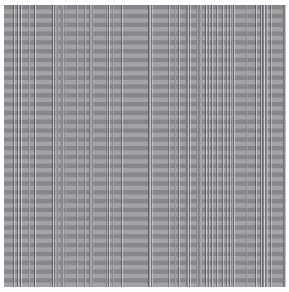
(a) One



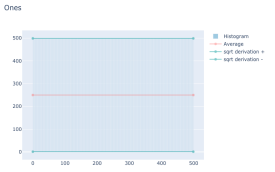
(b) 8 %



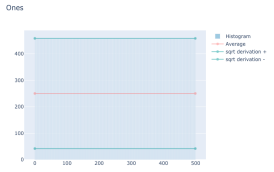
(c) 50 %



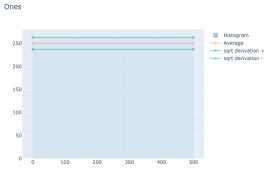
(d) 87 %



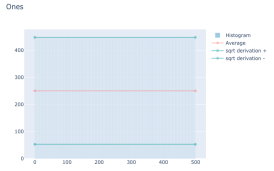
(a) One



(b) 8 %



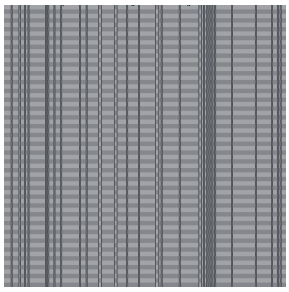
(c) 50 %



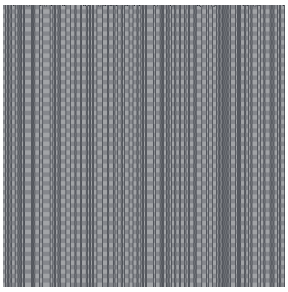
(d) 87 %

3.3.23. 33

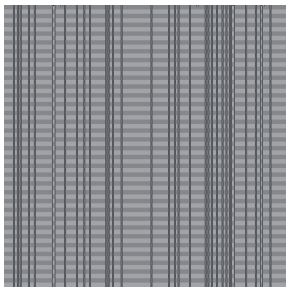
(a) One



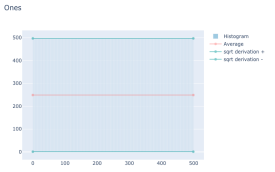
(b) 8 %



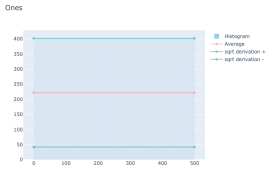
(c) 50 %



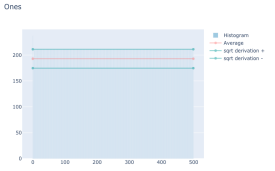
(d) 87 %



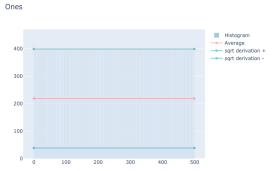
(a) One



(b) 8 %

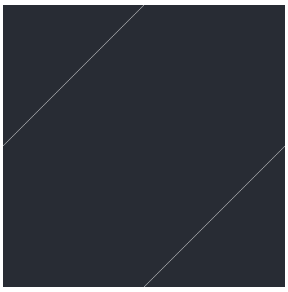


(c) 50 %



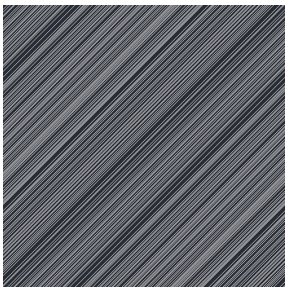
(d) 87 %

3.3.24. 34



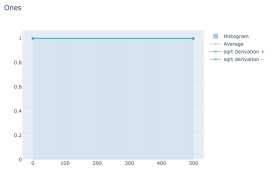
(a) One

(b) 8 %

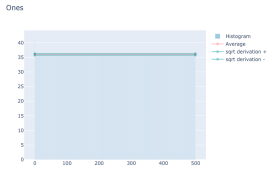


(c) 50 %

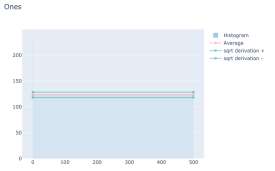
(d) 87 %



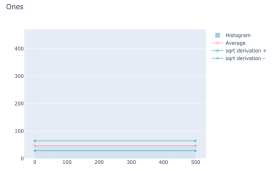
(a) One



(b) 8 %

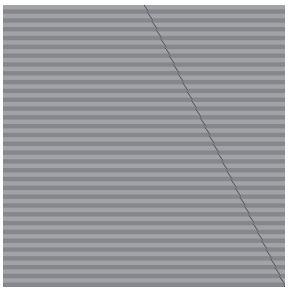


(c) 50 %

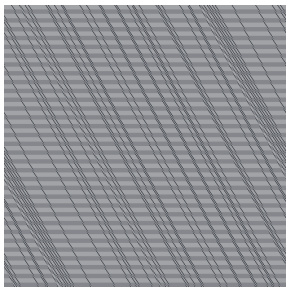


(d) 87 %

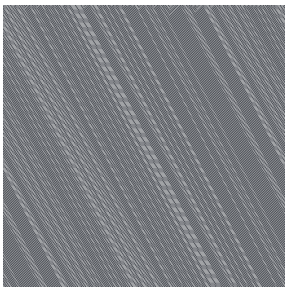
3.3.25. 35



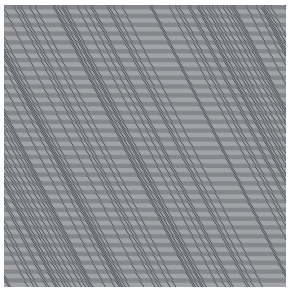
(a) One



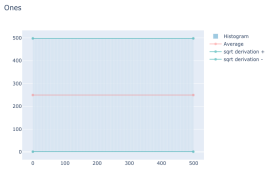
(b) 8 %



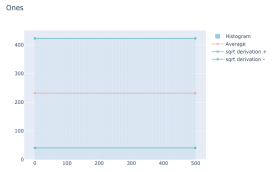
(c) 50 %



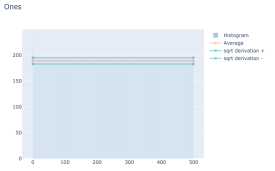
(d) 87 %



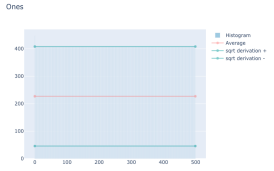
(a) One



(b) 8 %

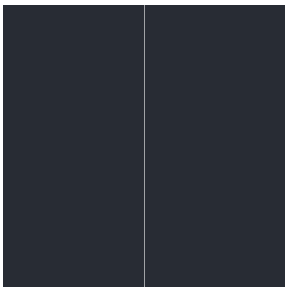


(c) 50 %

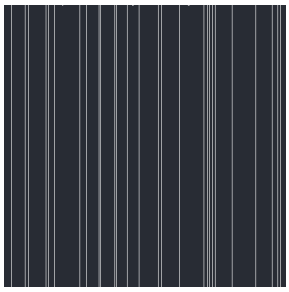


(d) 87 %

3.3.26. 36



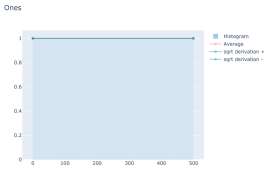
(a) One



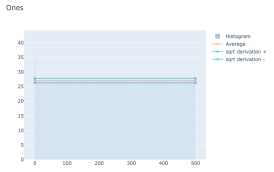
(b) 8 %

(c) 50 %

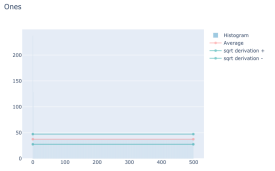
(d) 87 %



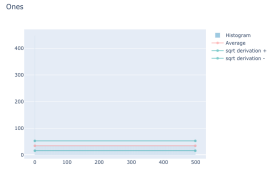
(a) One



(b) 8 %

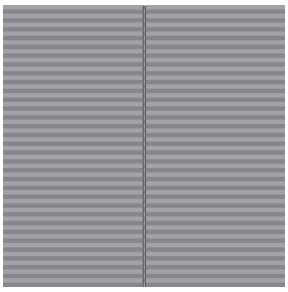


(c) 50 %

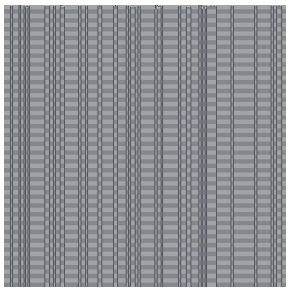


(d) 87 %

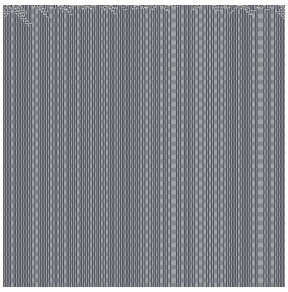
3.3.27. 37



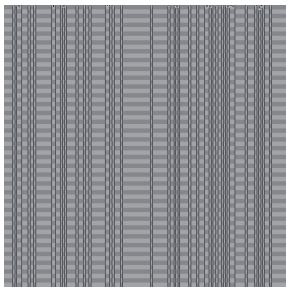
(a) One



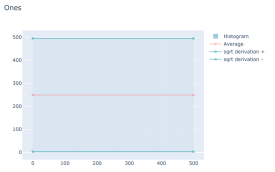
(b) 8 %



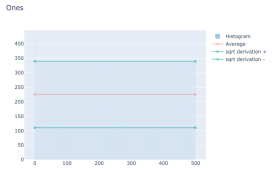
(c) 50 %



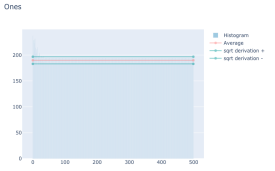
(d) 87 %



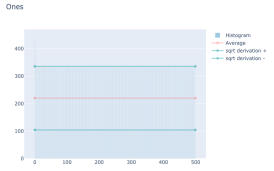
(a) One



(b) 8 %

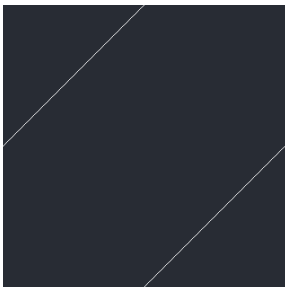


(c) 50 %



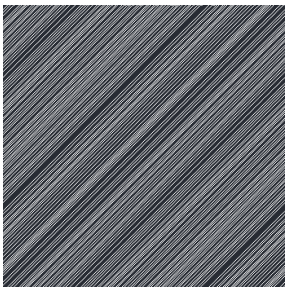
(d) 87 %

3.3.28. 38

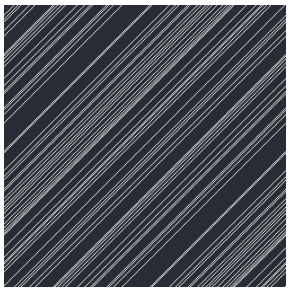


(a) One

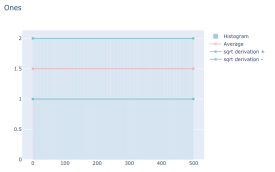
(b) 8 %



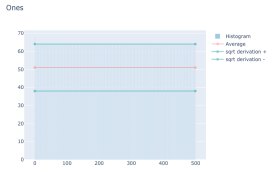
(c) 50 %



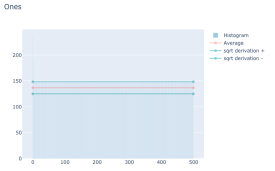
(d) 87 %



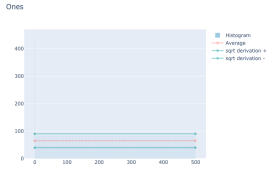
(a) One



(b) 8 %

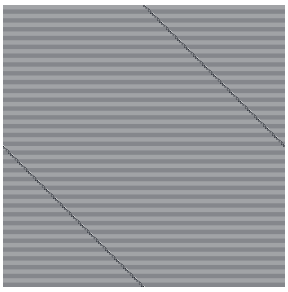


(c) 50 %

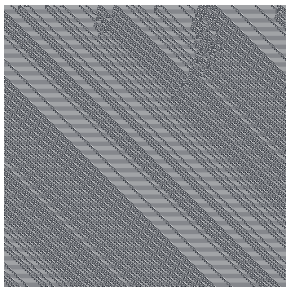


(d) 87 %

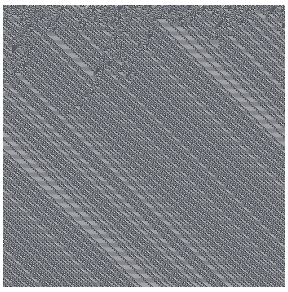
3.3.29. 41



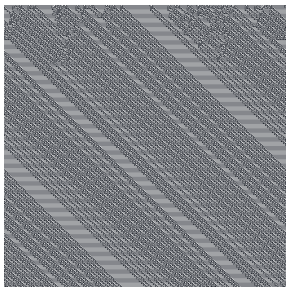
(a) One



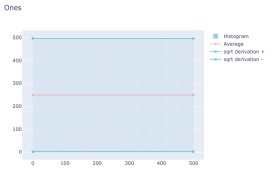
(b) 8 %



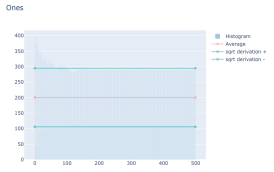
(c) 50 %



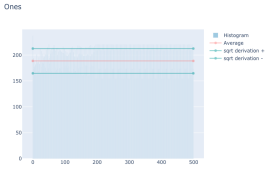
(d) 87 %



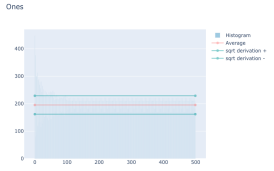
(a) One



(b) 8 %

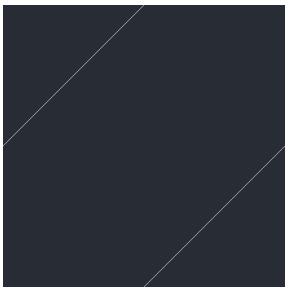


(c) 50 %



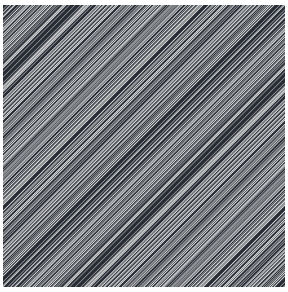
(d) 87 %

3.3.30. 42

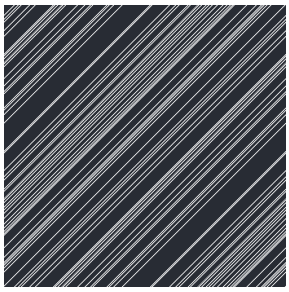


(a) One

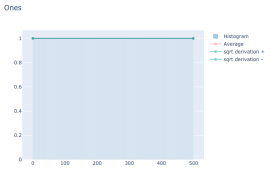
(b) 8 %



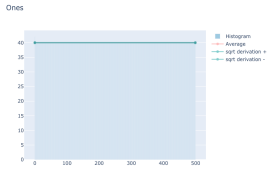
(c) 50 %



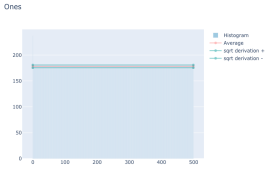
(d) 87 %



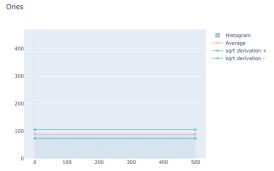
(a) One



(b) 8 %

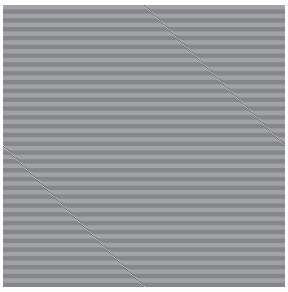


(c) 50 %

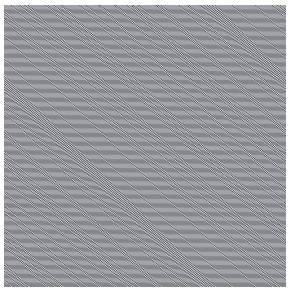


(d) 87 %

3.3.31. 43

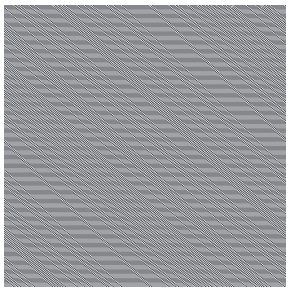


(a) One

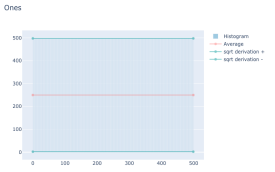


(b) 8 %

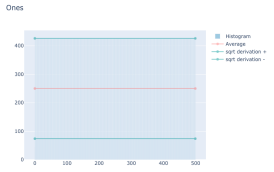
(c) 50 %



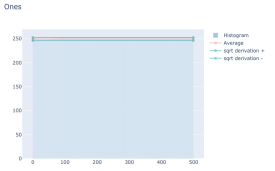
(d) 87 %



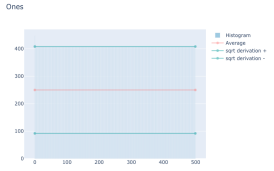
(a) One



(b) 8 %

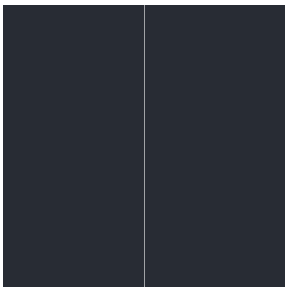


(c) 50 %



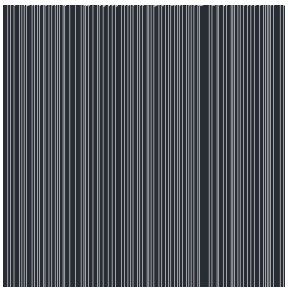
(d) 87 %

3.3.32. 44



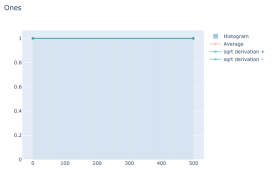
(a) One

(b) 8 %

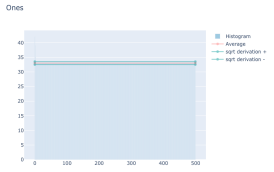


(c) 50 %

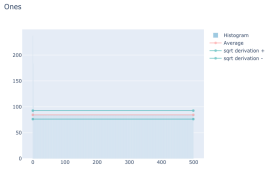
(d) 87 %



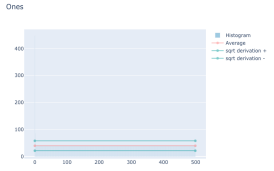
(a) One



(b) 8 %

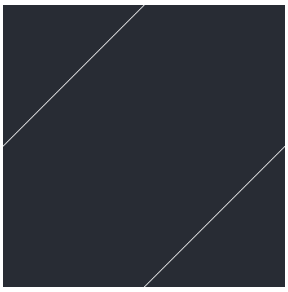


(c) 50 %



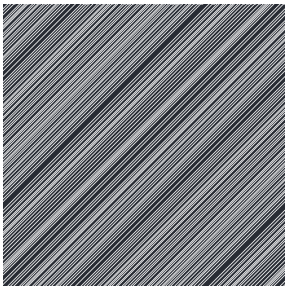
(d) 87 %

3.3.33. 46

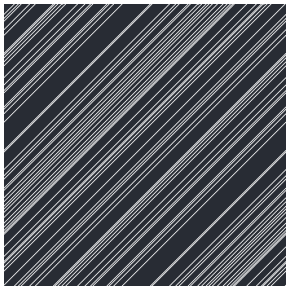


(a) One

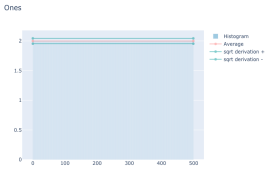
(b) 8 %



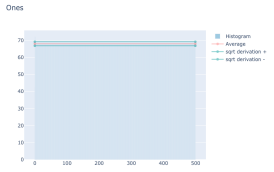
(c) 50 %



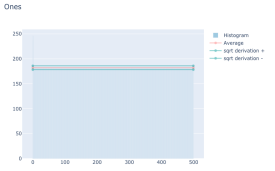
(d) 87 %



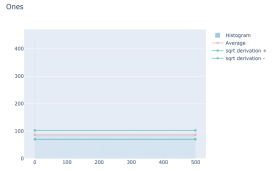
(a) One



(b) 8 %

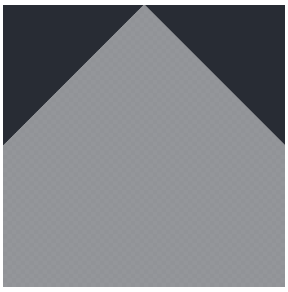


(c) 50 %

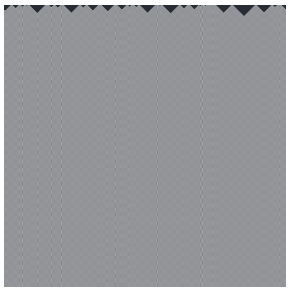


(d) 87 %

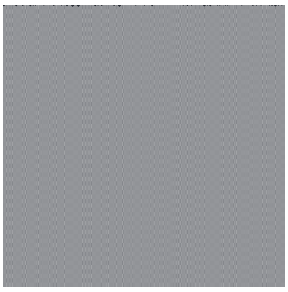
3.3.34. 50



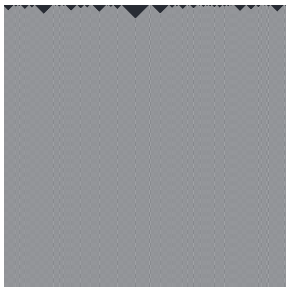
(a) One



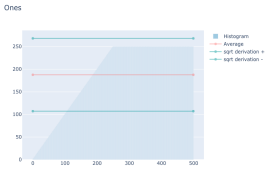
(b) 8 %



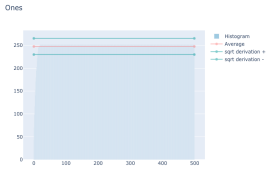
(c) 50 %



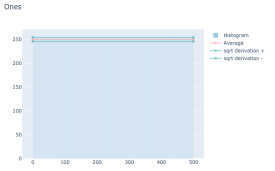
(d) 87 %



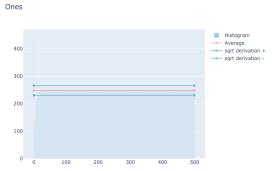
(a) One



(b) 8 %



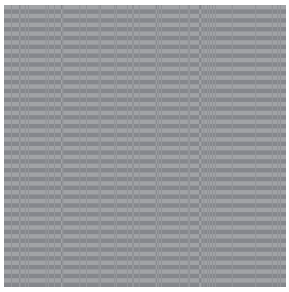
(c) 50 %



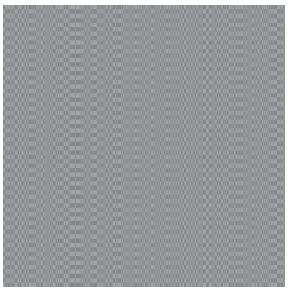
(d) 87 %

3.3.35. 51

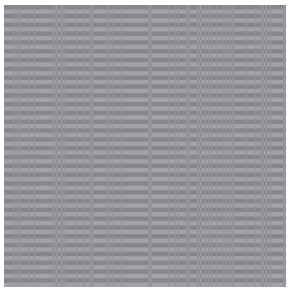
(a) One



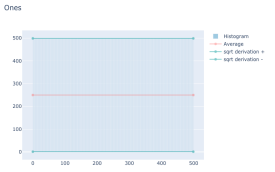
(b) 8 %



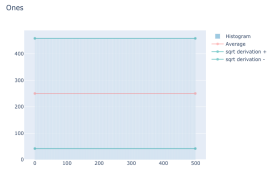
(c) 50 %



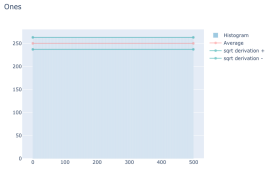
(d) 87 %



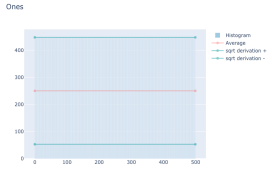
(a) One



(b) 8 %



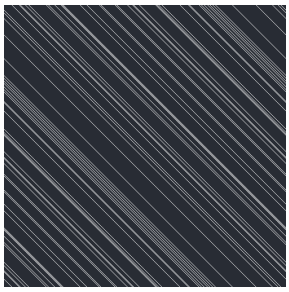
(c) 50 %



(d) 87 %

3.3.36. 56

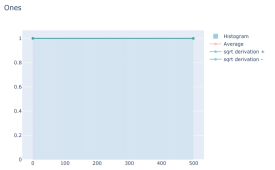
(a) One



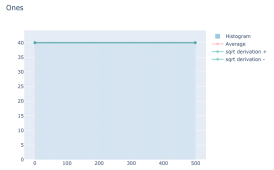
(b) 8 %

(c) 50 %

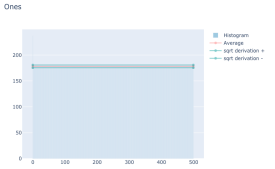
(d) 87 %



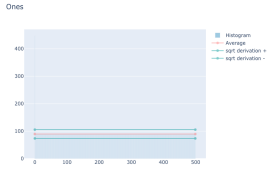
(a) One



(b) 8 %

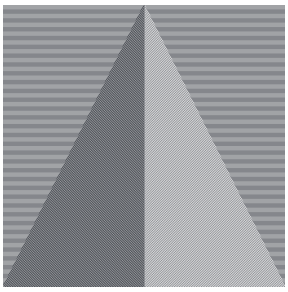


(c) 50 %

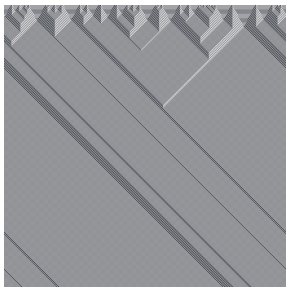


(d) 87 %

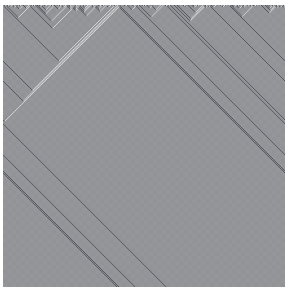
3.3.37. 57



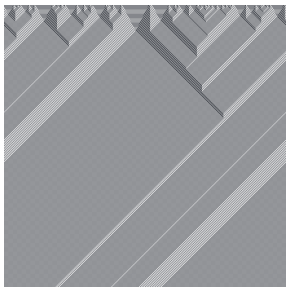
(a) One



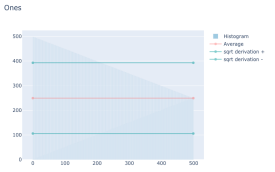
(b) 8 %



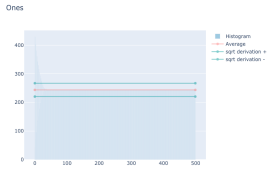
(c) 50 %



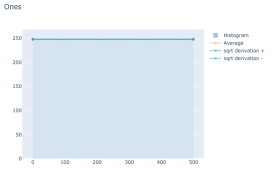
(d) 87 %



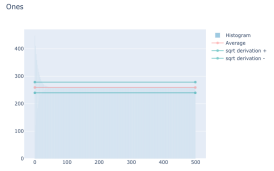
(a) One



(b) 8 %

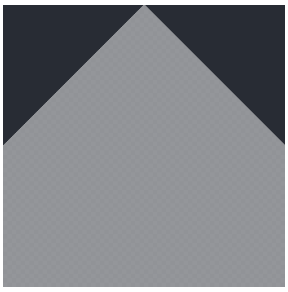


(c) 50 %



(d) 87 %

3.3.38. 58

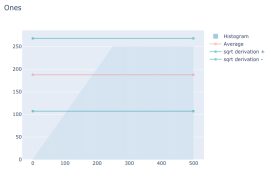


(a) One

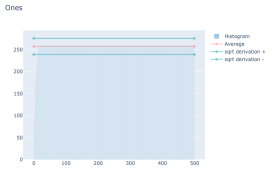
(b) 8 %

(c) 50 %

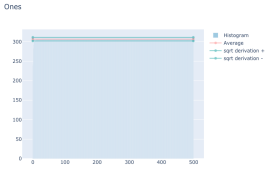
(d) 87 %



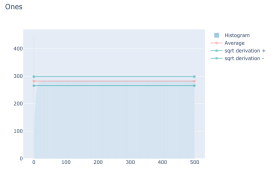
(a) One



(b) 8 %

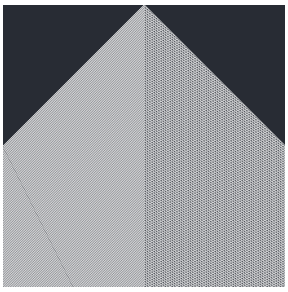


(c) 50 %

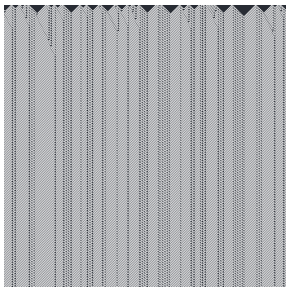


(d) 87 %

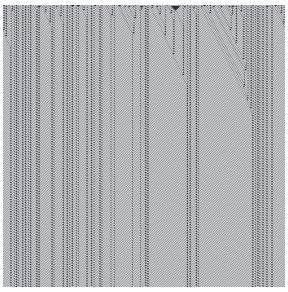
3.3.39. 62



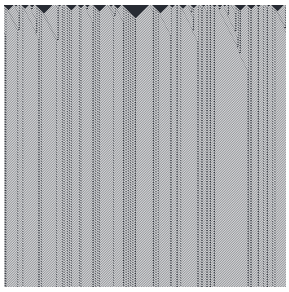
(a) One



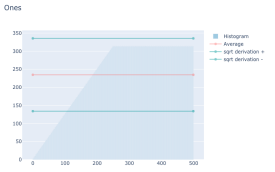
(b) 8 %



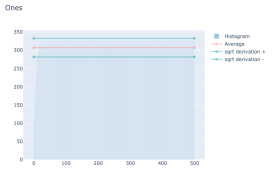
(c) 50 %



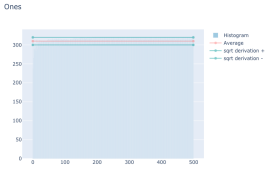
(d) 87 %



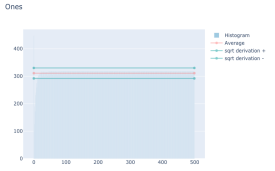
(a) One



(b) 8 %



(c) 50 %



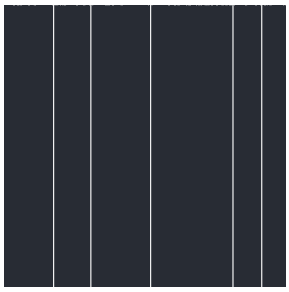
(d) 87 %

3.3.40. 72

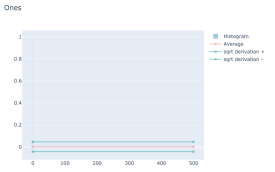
(a) One

(b) 8 %

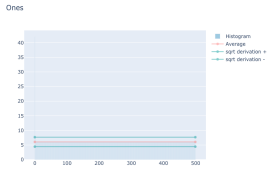
(c) 50 %



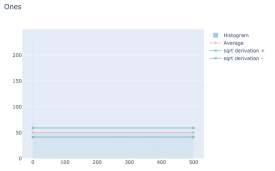
(d) 87 %



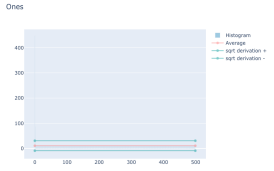
(a) One



(b) 8 %

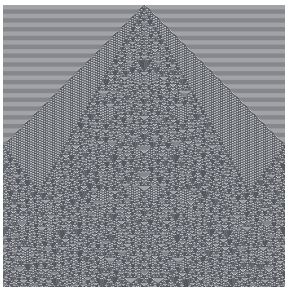


(c) 50 %

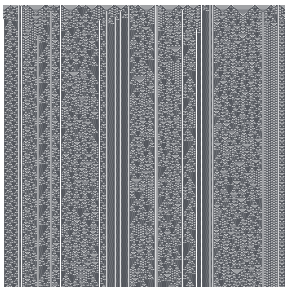


(d) 87 %

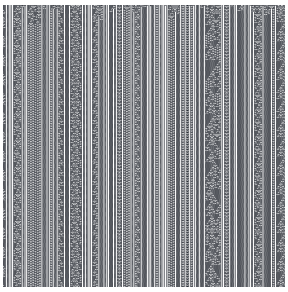
3.3.41. 73



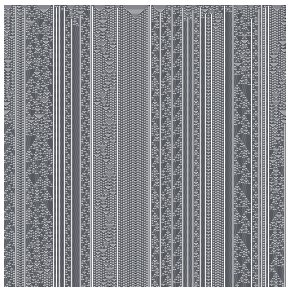
(a) One



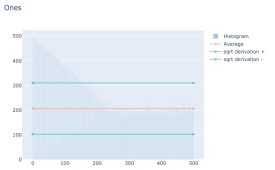
(b) 8 %



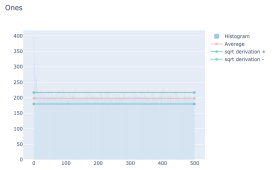
(c) 50 %



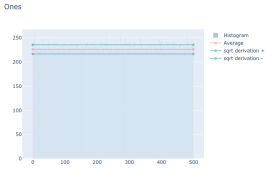
(d) 87 %



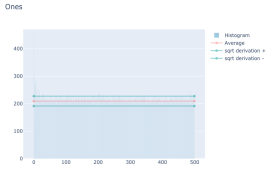
(a) One



(b) 8 %

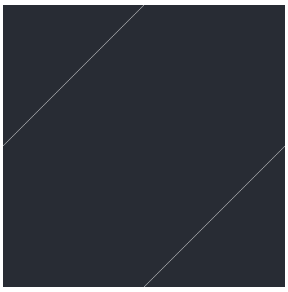


(c) 50 %

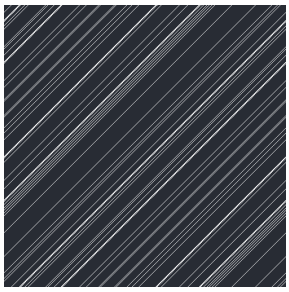


(d) 87 %

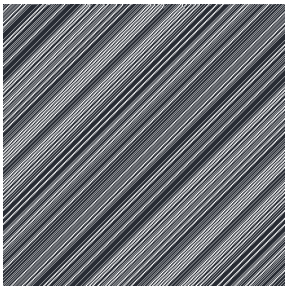
3.3.42. 74



(a) One

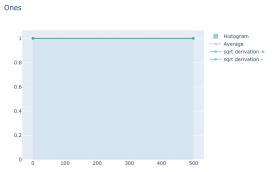


(b) 8 %

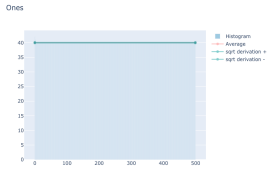


(c) 50 %

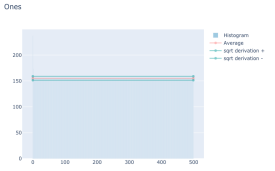
(d) 87 %



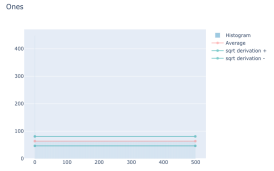
(a) One



(b) 8 %

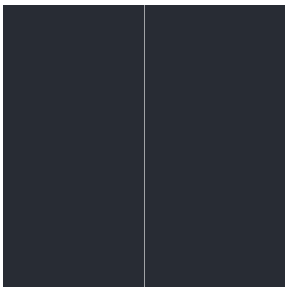


(c) 50 %



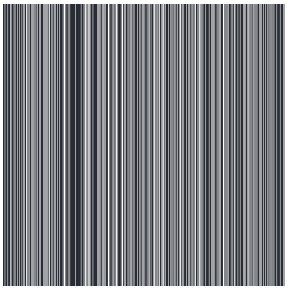
(d) 87 %

3.3.43. 76



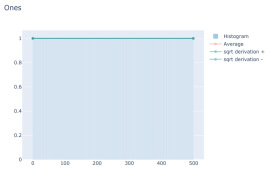
(a) One

(b) 8 %

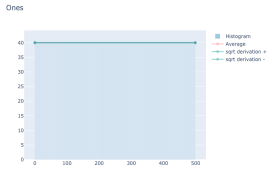


(c) 50 %

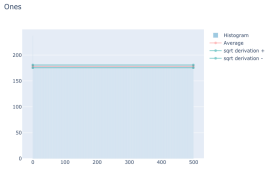
(d) 87 %



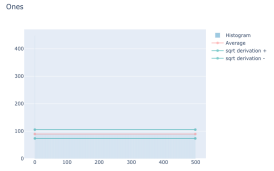
(a) One



(b) 8 %

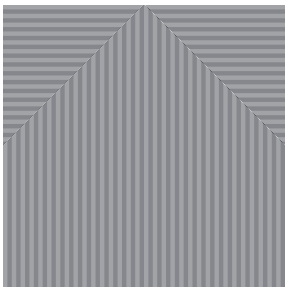


(c) 50 %

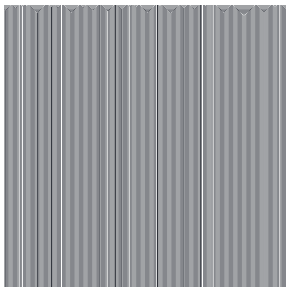


(d) 87 %

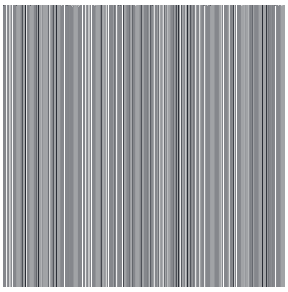
3.3.44. 77



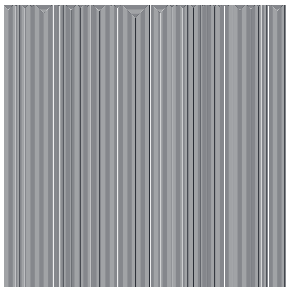
(a) One



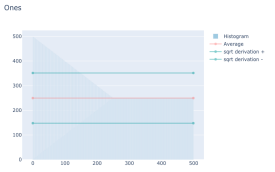
(b) 8 %



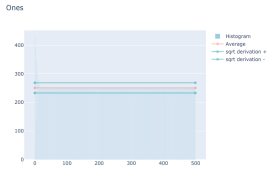
(c) 50 %



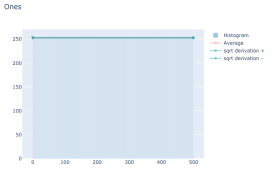
(d) 87 %



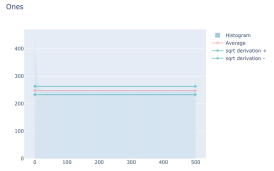
(a) One



(b) 8 %

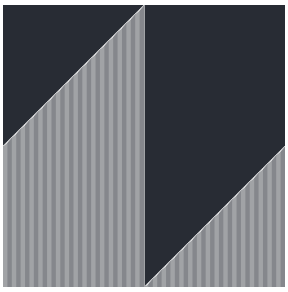


(c) 50 %

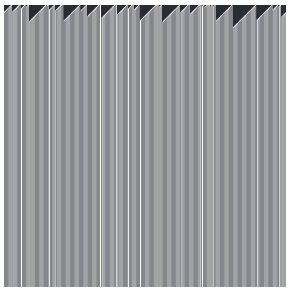


(d) 87 %

3.3.45. 78

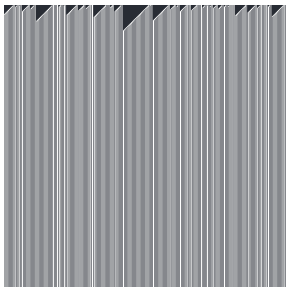


(a) One

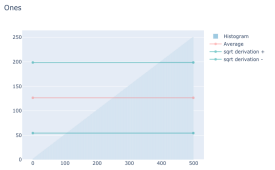


(b) 8 %

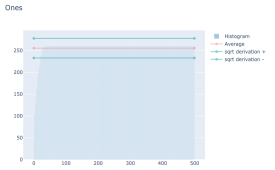
(c) 50 %



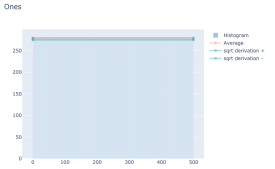
(d) 87 %



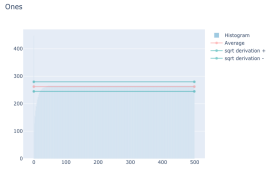
(a) One



(b) 8 %

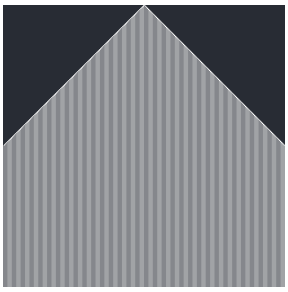


(c) 50 %



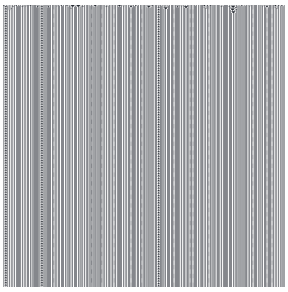
(d) 87 %

3.3.46. 94

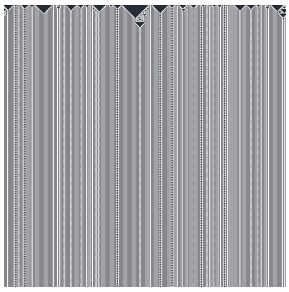


(a) One

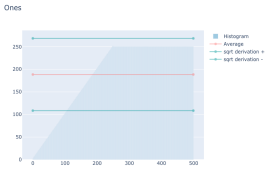
(b) 8 %



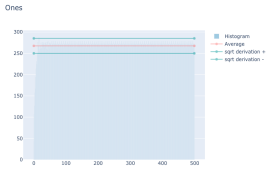
(c) 50 %



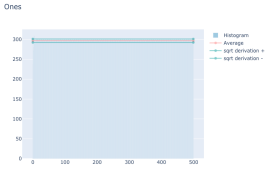
(d) 87 %



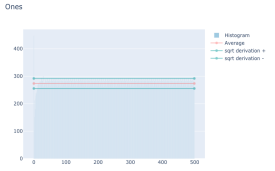
(a) One



(b) 8 %



(c) 50 %



(d) 87 %

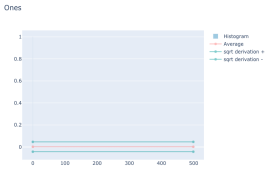
3.3.47. 104

(a) One

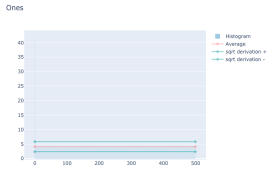
(b) 8 %

(c) 50 %

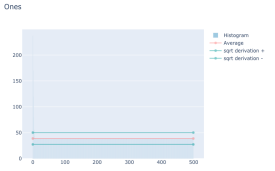
(d) 87 %



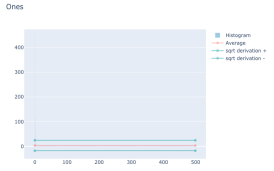
(a) One



(b) 8 %

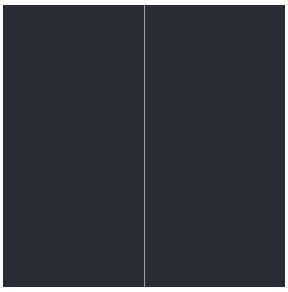


(c) 50 %



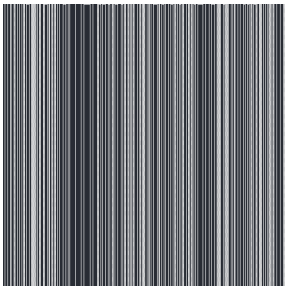
(d) 87 %

3.3.48. 108



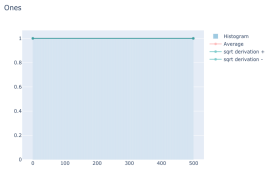
(a) One

(b) 8 %

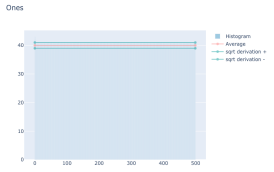


(c) 50 %

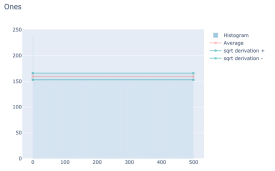
(d) 87 %



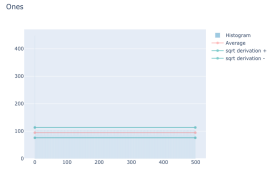
(a) One



(b) 8 %

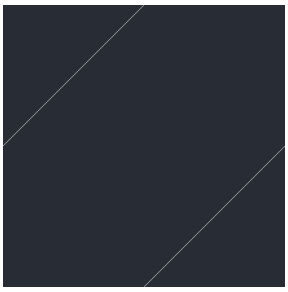


(c) 50 %



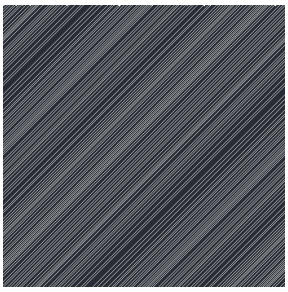
(d) 87 %

3.3.49. 130

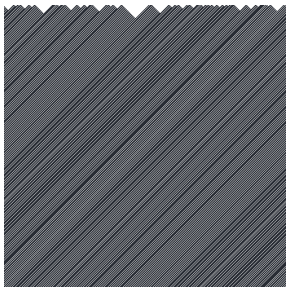


(a) One

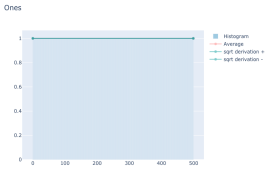
(b) 8 %



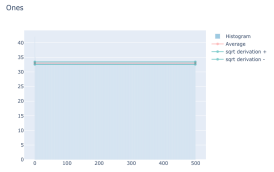
(c) 50 %



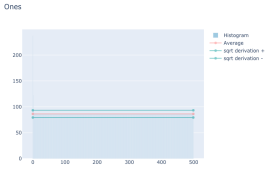
(d) 87 %



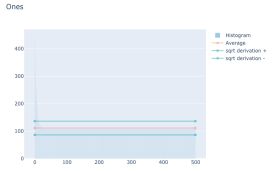
(a) One



(b) 8 %

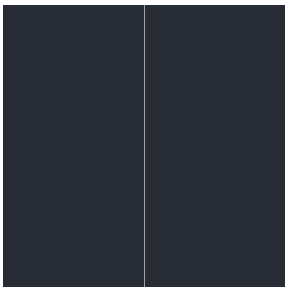


(c) 50 %

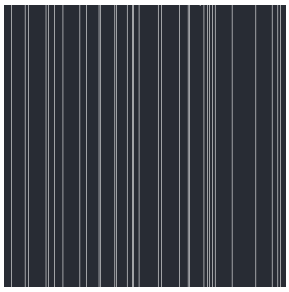


(d) 87 %

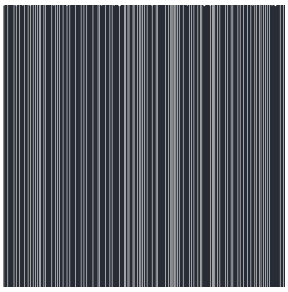
3.3.50. 132



(a) One

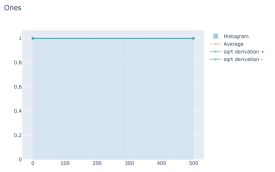


(b) 8 %

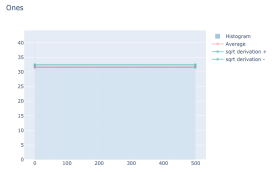


(c) 50 %

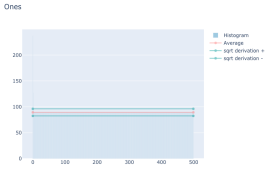
(d) 87 %



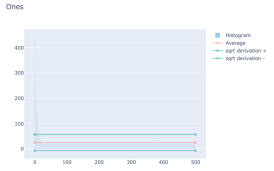
(a) One



(b) 8 %

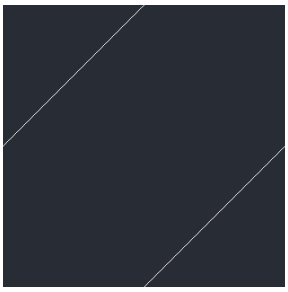


(c) 50 %



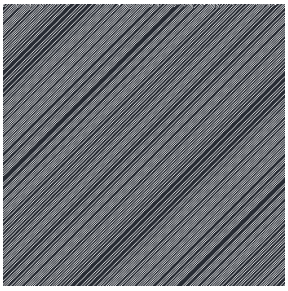
(d) 87 %

3.3.51. 134

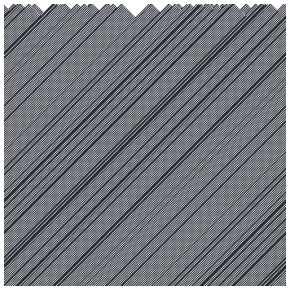


(a) One

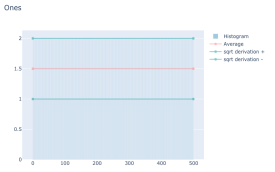
(b) 8 %



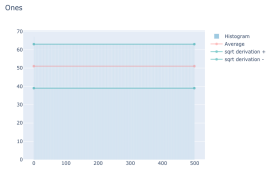
(c) 50 %



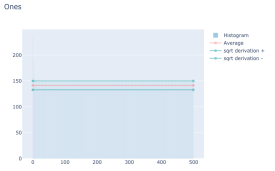
(d) 87 %



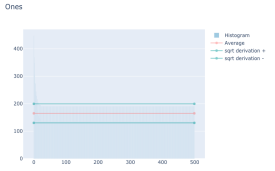
(a) One



(b) 8 %

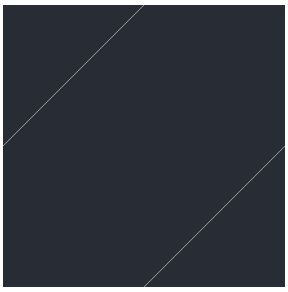


(c) 50 %



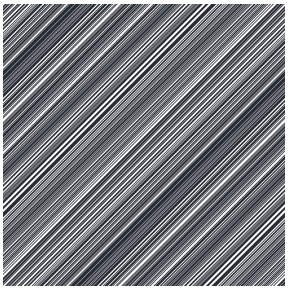
(d) 87 %

3.3.52. 138

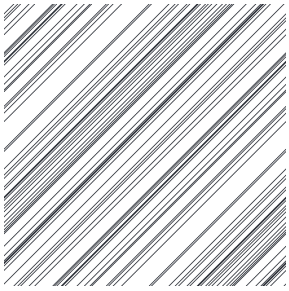


(a) One

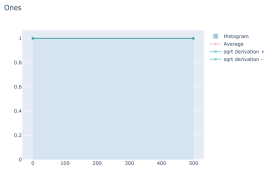
(b) 8 %



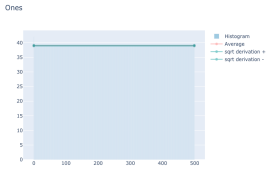
(c) 50 %



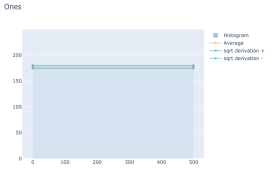
(d) 87 %



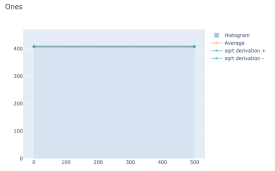
(a) One



(b) 8 %

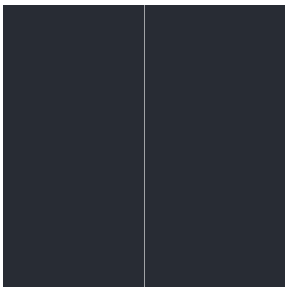


(c) 50 %



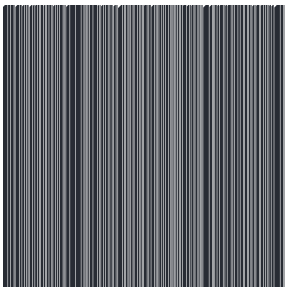
(d) 87 %

3.3.53. 140



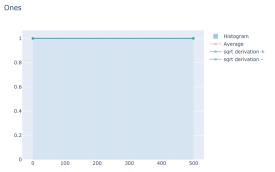
(a) One

(b) 8 %

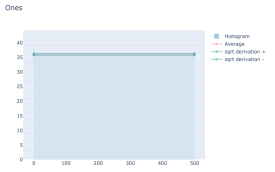


(c) 50 %

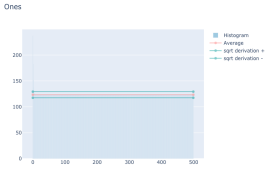
(d) 87 %



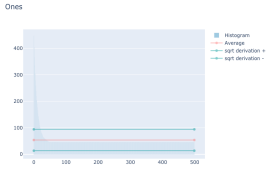
(a) One



(b) 8 %

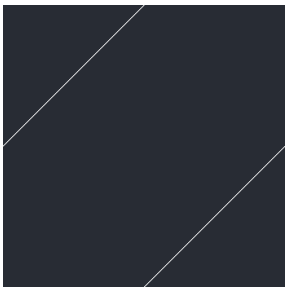


(c) 50 %



(d) 87 %

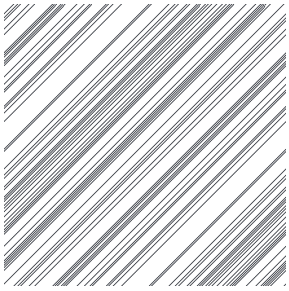
3.3.54. 142



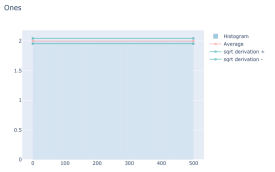
(a) One

(b) 8 %

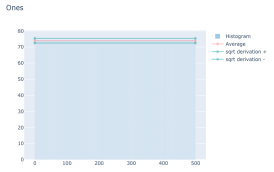
(c) 50 %



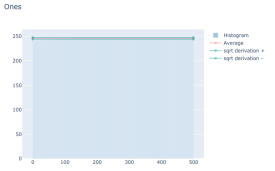
(d) 87 %



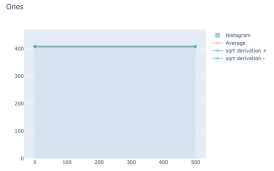
(a) One



(b) 8 %



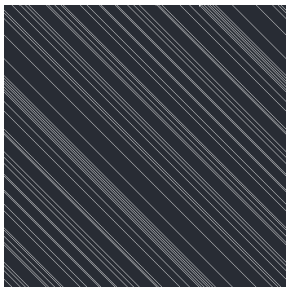
(c) 50 %



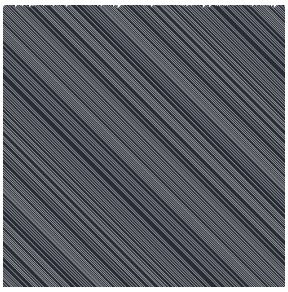
(d) 87 %

3.3.55. 152

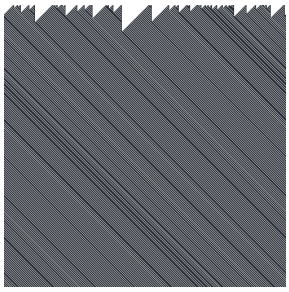
(a) One



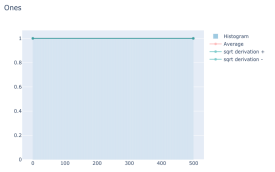
(b) 8 %



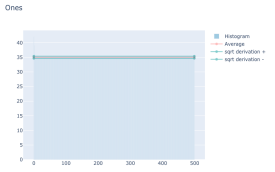
(c) 50 %



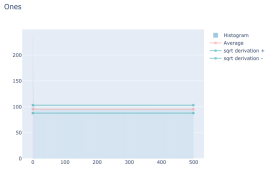
(d) 87 %



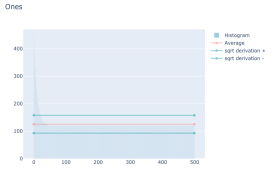
(a) One



(b) 8 %

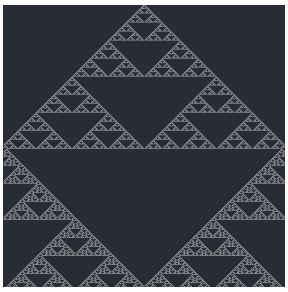


(c) 50 %

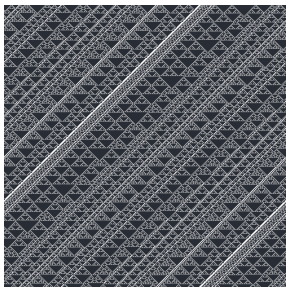


(d) 87 %

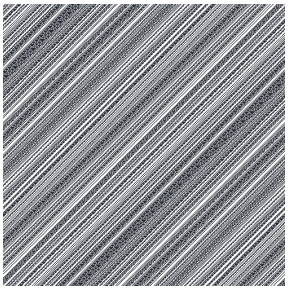
3.3.56. 154



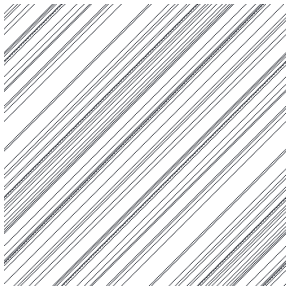
(a) One



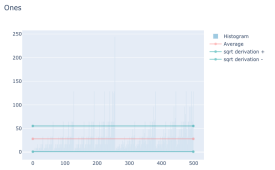
(b) 8 %



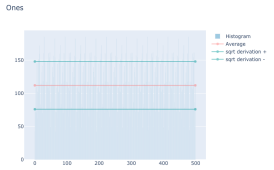
(c) 50 %



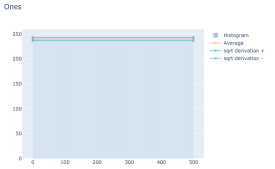
(d) 87 %



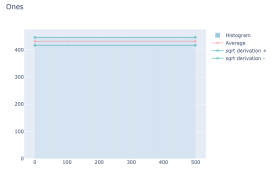
(a) One



(b) 8 %



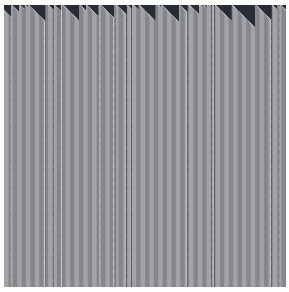
(c) 50 %



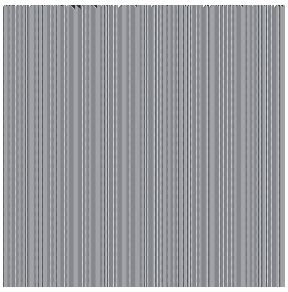
(d) 87 %

3.3.57. 156

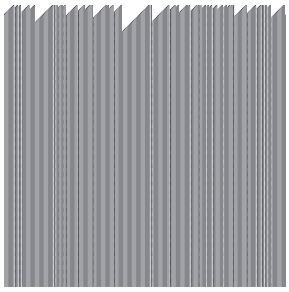
(a) One



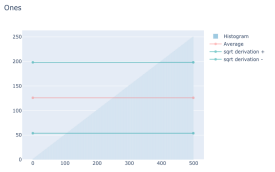
(b) 8 %



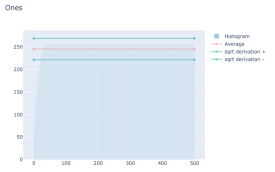
(c) 50 %



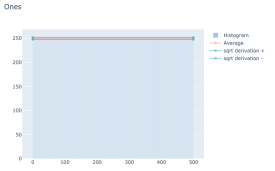
(d) 87 %



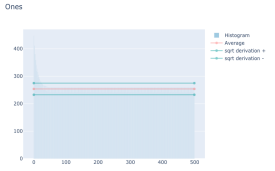
(a) One



(b) 8 %

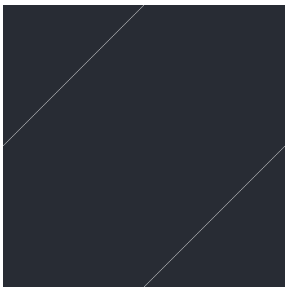


(c) 50 %



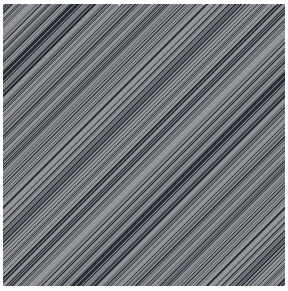
(d) 87 %

3.3.58. 162

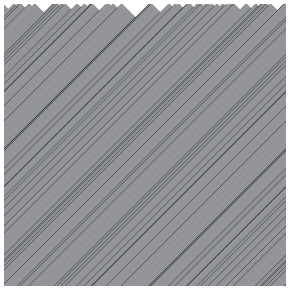


(a) One

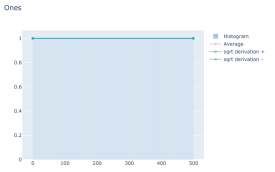
(b) 8 %



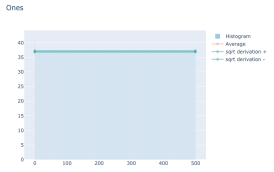
(c) 50 %



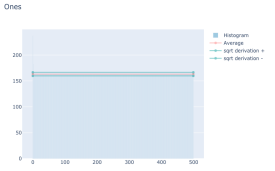
(d) 87 %



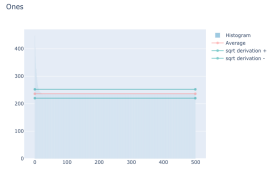
(a) One



(b) 8 %

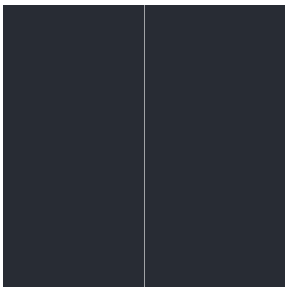


(c) 50 %

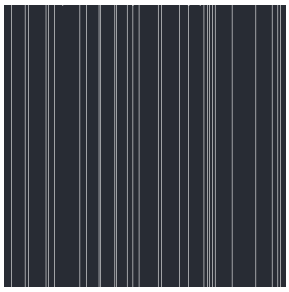


(d) 87 %

3.3.59. 164



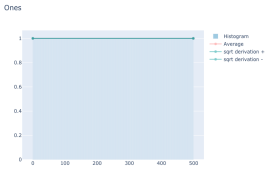
(a) One



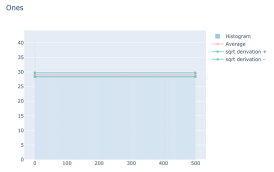
(b) 8 %

(c) 50 %

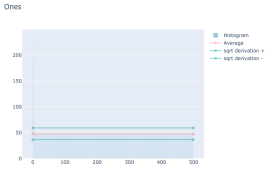
(d) 87 %



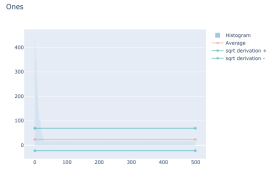
(a) One



(b) 8 %

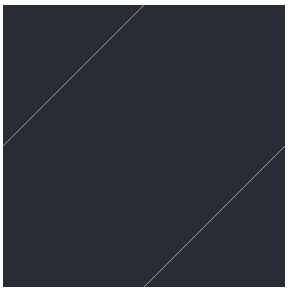


(c) 50 %



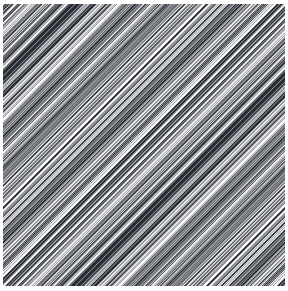
(d) 87 %

3.3.60. 170

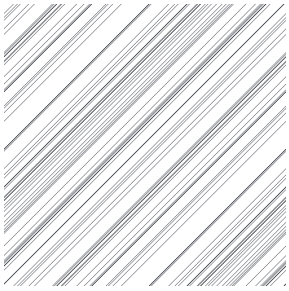


(a) One

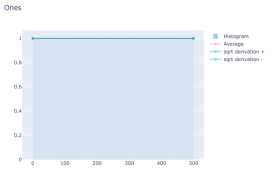
(b) 8 %



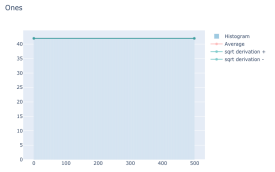
(c) 50 %



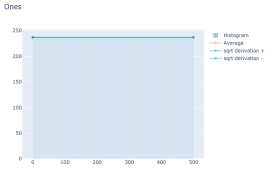
(d) 87 %



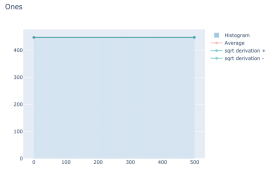
(a) One



(b) 8 %

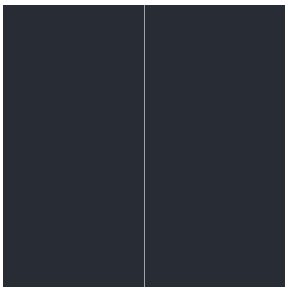


(c) 50 %



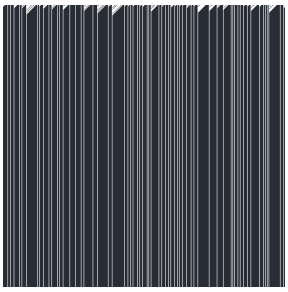
(d) 87 %

3.3.61. 172

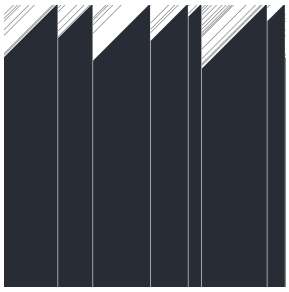


(a) One

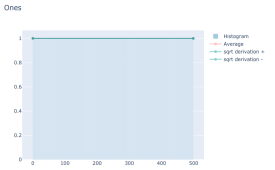
(b) 8 %



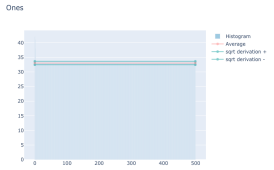
(c) 50 %



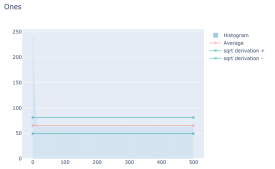
(d) 87 %



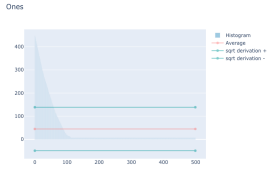
(a) One



(b) 8 %

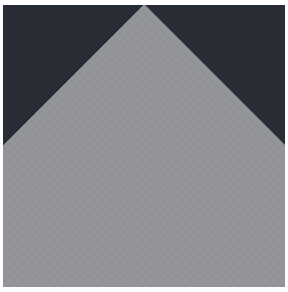


(c) 50 %

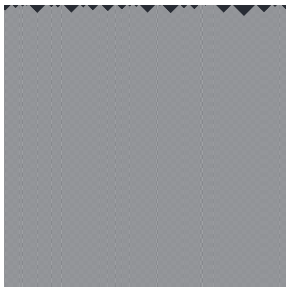


(d) 87 %

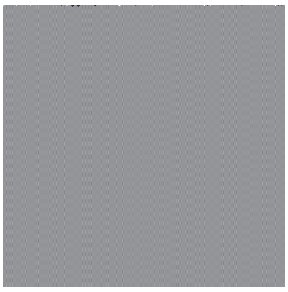
3.3.62. 178



(a) One

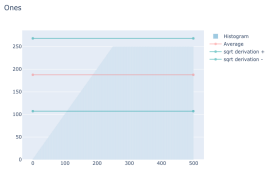


(b) 8 %

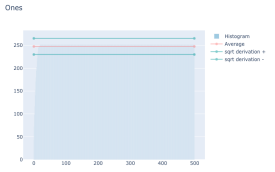


(c) 50 %

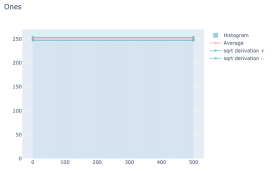
(d) 87 %



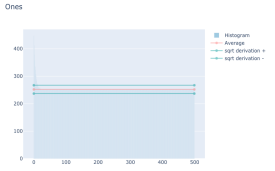
(a) One



(b) 8 %



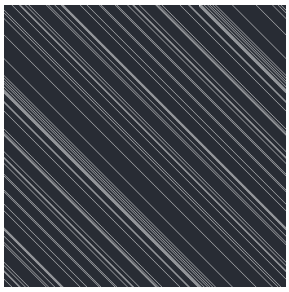
(c) 50 %



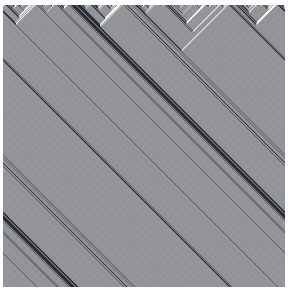
(d) 87 %

3.3.63. 184

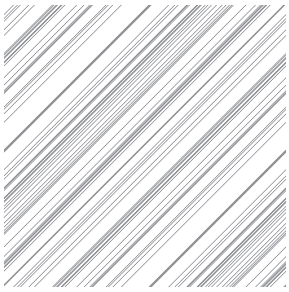
(a) One



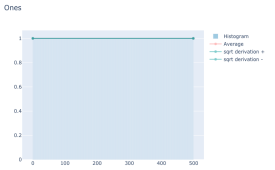
(b) 8 %



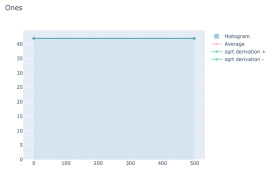
(c) 50 %



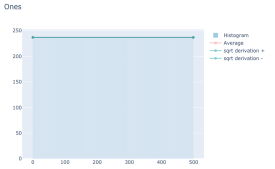
(d) 87 %



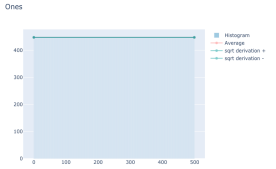
(a) One



(b) 8 %



(c) 50 %

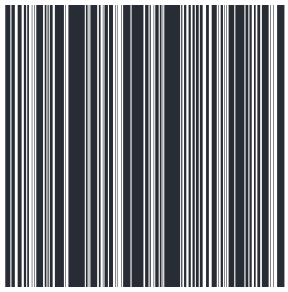


(d) 87 %

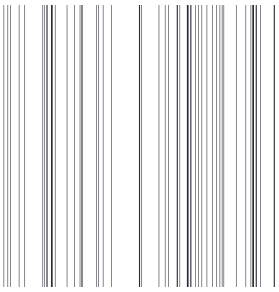
3.3.64. 200

(a) One

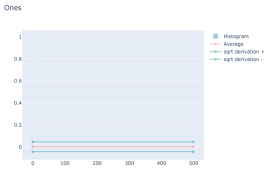
(b) 8 %



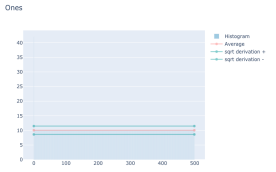
(c) 50 %



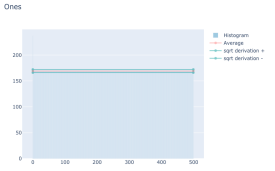
(d) 87 %



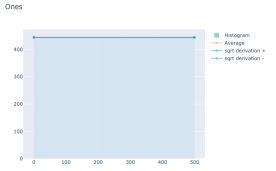
(a) One



(b) 8 %

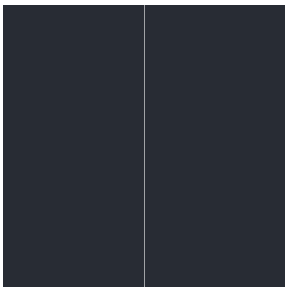


(c) 50 %



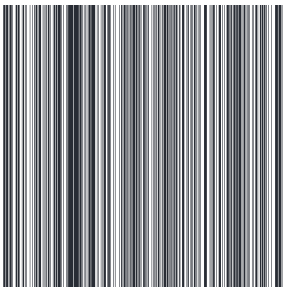
(d) 87 %

3.3.65. 204

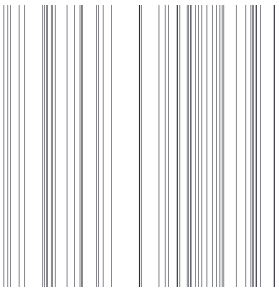


(a) One

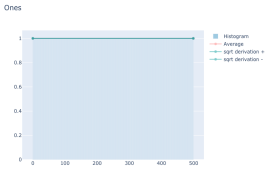
(b) 8 %



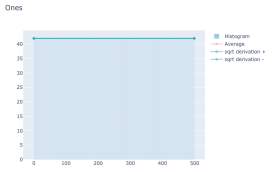
(c) 50 %



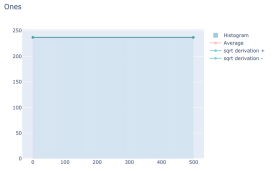
(d) 87 %



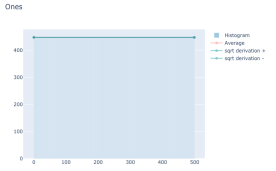
(a) One



(b) 8 %



(c) 50 %

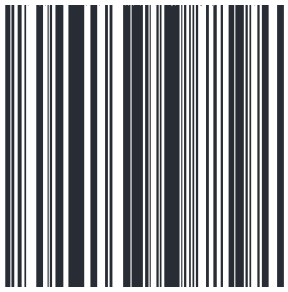


(d) 87 %

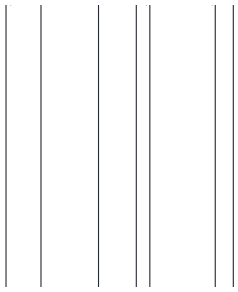
3.3.66. 232

(a) One

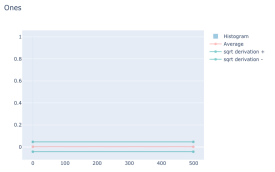
(b) 8 %



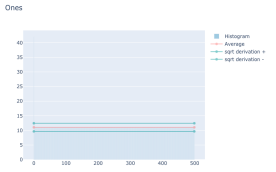
(c) 50 %



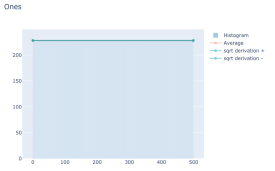
(d) 87 %



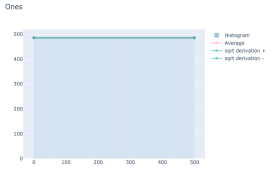
(a) One



(b) 8 %



(c) 50 %



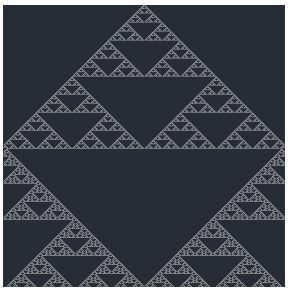
(d) 87 %

3.4. W3

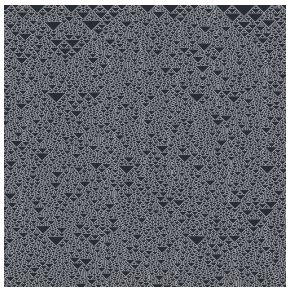
Casi todas las configuraciones iniciales conducen a un comportamiento esencialmente aleatorio o caoticos Casi todos los patrones iniciales evolucionan de forma pseudo-aleatoria o caótica. Las estructuras estables que aparecen son destruidas rápidamente por el ruido circundante. Los cambios locales en el patrón inicial tienden a propagarse indefinidamente.

Lo que buscamos aqui es un histograma que no se estabilize en una cantidad constante de unos.

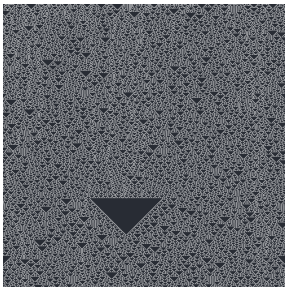
3.4.1. 18



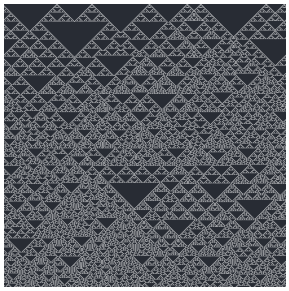
(a) One



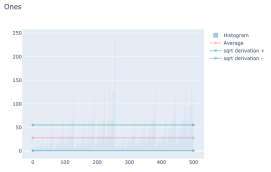
(b) 8 %



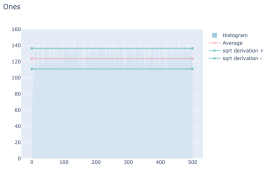
(c) 50 %



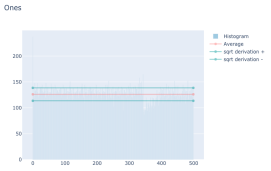
(d) 87 %



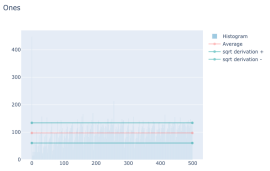
(a) One



(b) 8 %

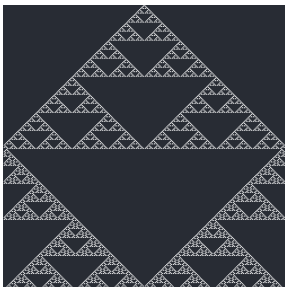


(c) 50 %

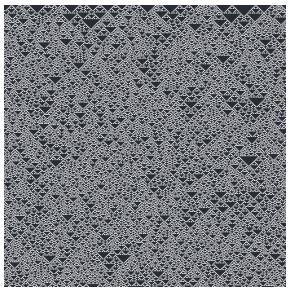


(d) 87 %

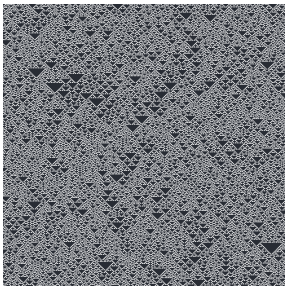
3.4.2. 22



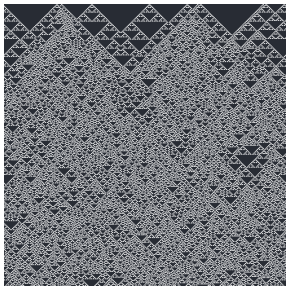
(a) One



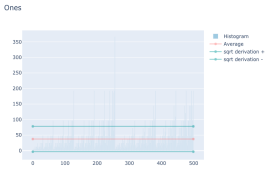
(b) 8 %



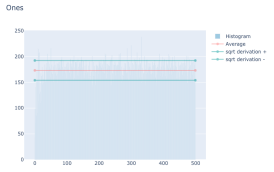
(c) 50 %



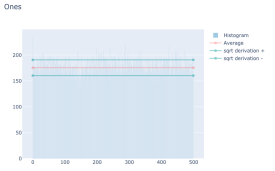
(d) 87 %



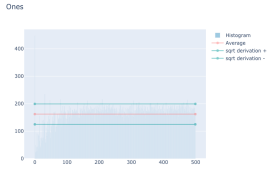
(a) One



(b) 8 %

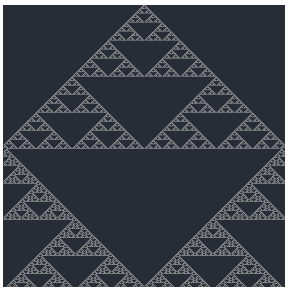


(c) 50 %

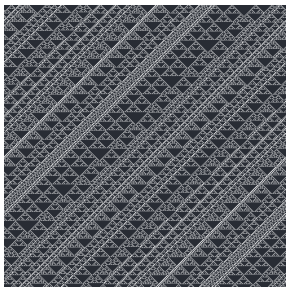


(d) 87 %

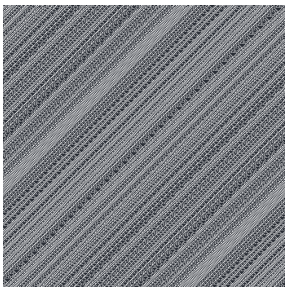
3.4.3. 26



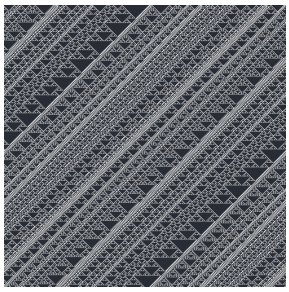
(a) One



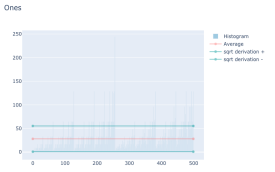
(b) 8 %



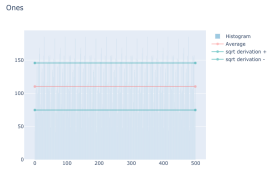
(c) 50 %



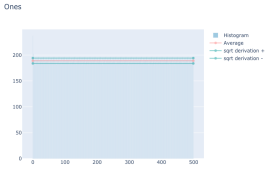
(d) 87 %



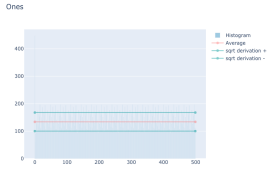
(a) One



(b) 8 %

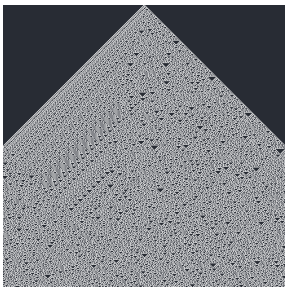


(c) 50 %

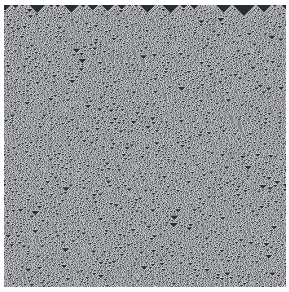


(d) 87 %

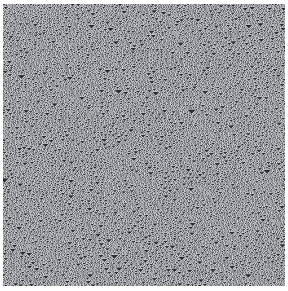
3.4.4. 30



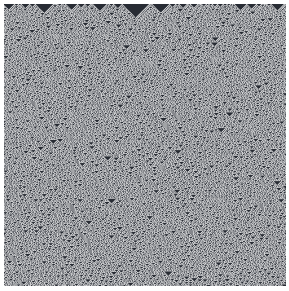
(a) One



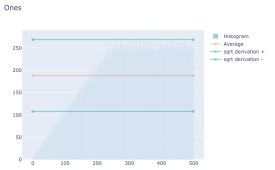
(b) 8 %



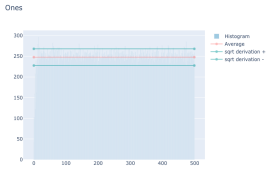
(c) 50 %



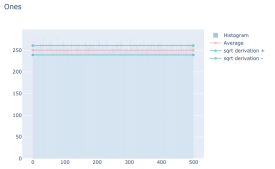
(d) 87 %



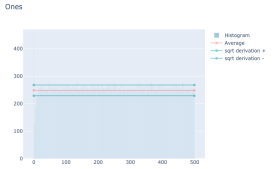
(a) One



(b) 8 %

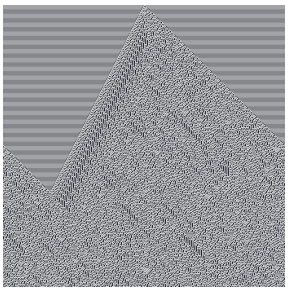


(c) 50 %

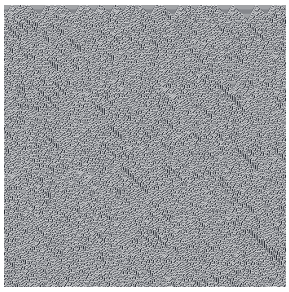


(d) 87 %

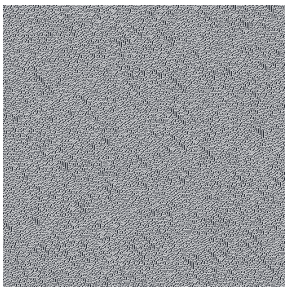
3.4.5. 45



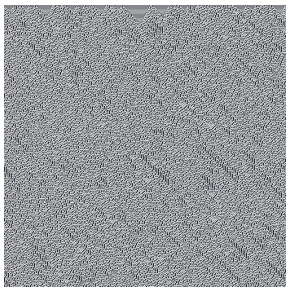
(a) One



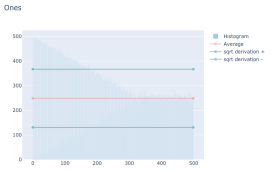
(b) 8 %



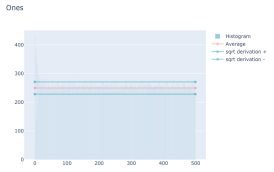
(c) 50 %



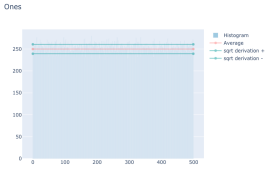
(d) 87 %



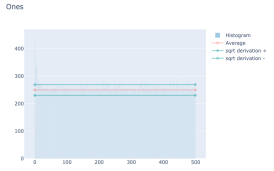
(a) One



(b) 8 %

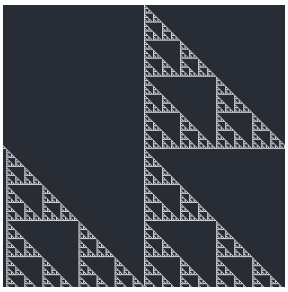


(c) 50 %

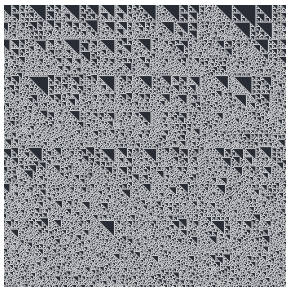


(d) 87 %

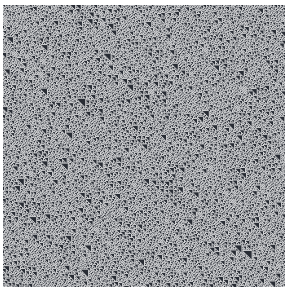
3.4.6. 60



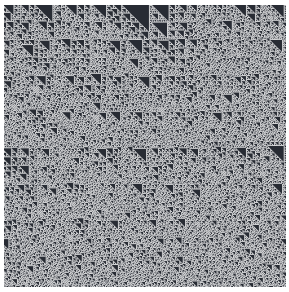
(a) One



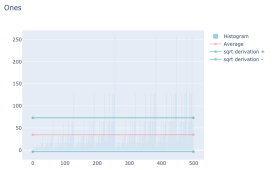
(b) 8 %



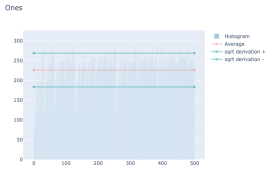
(c) 50 %



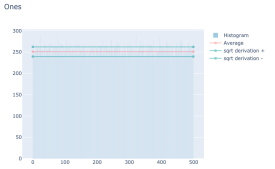
(d) 87 %



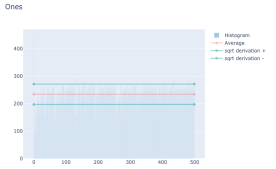
(a) One



(b) 8 %

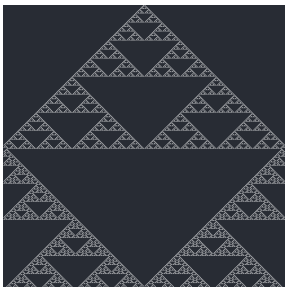


(c) 50 %

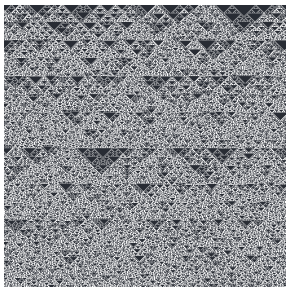


(d) 87 %

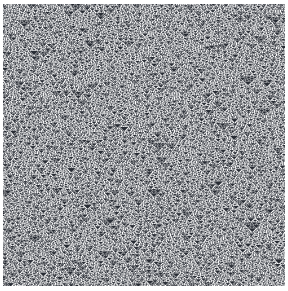
3.4.7. 90



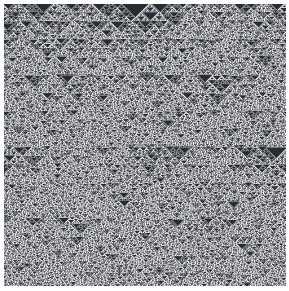
(a) One



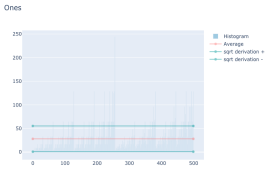
(b) 8 %



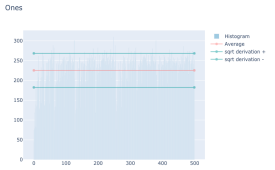
(c) 50 %



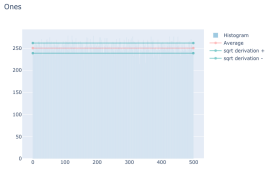
(d) 87 %



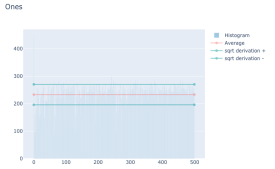
(a) One



(b) 8 %

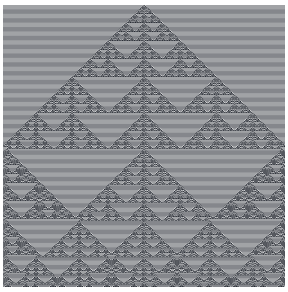


(c) 50 %

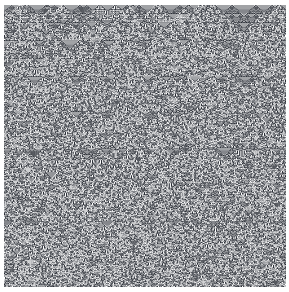


(d) 87 %

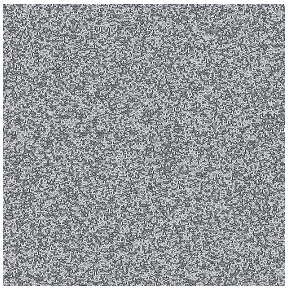
3.4.8. 105



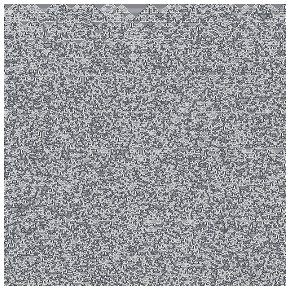
(a) One



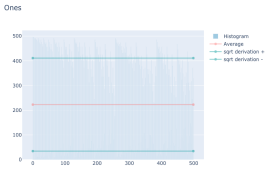
(b) 8 %



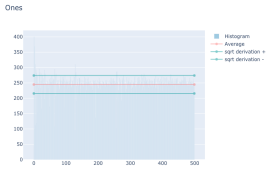
(c) 50 %



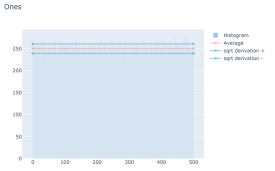
(d) 87 %



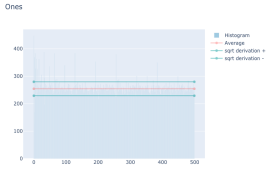
(a) One



(b) 8 %

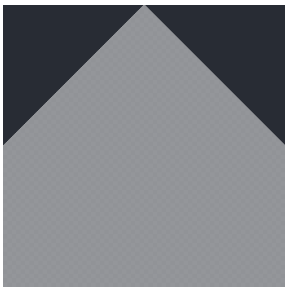


(c) 50 %

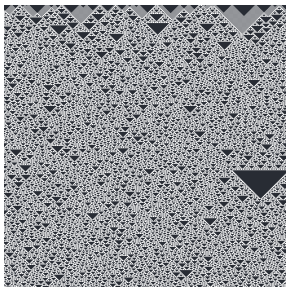


(d) 87 %

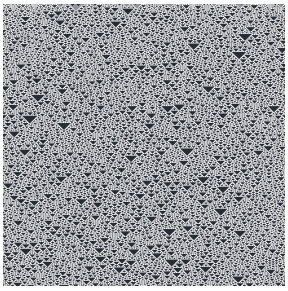
3.4.9. 122



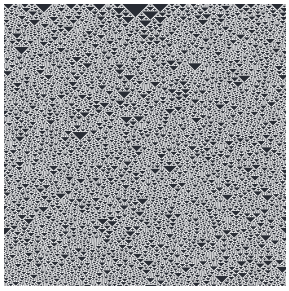
(a) One



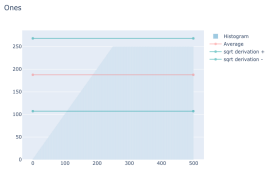
(b) 8 %



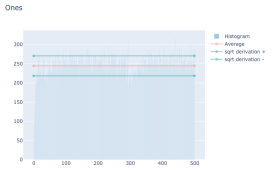
(c) 50 %



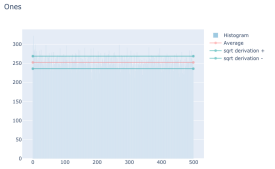
(d) 87 %



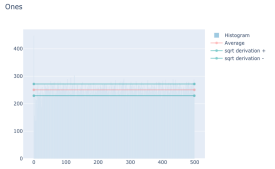
(a) One



(b) 8 %

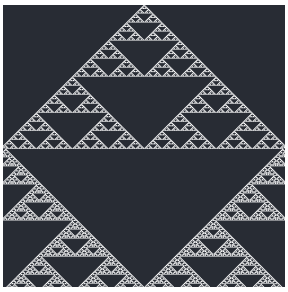


(c) 50 %

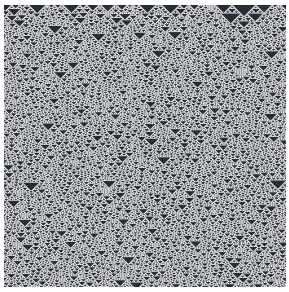


(d) 87 %

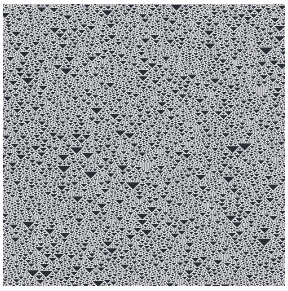
3.4.10. 126



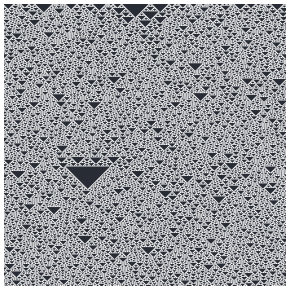
(a) One



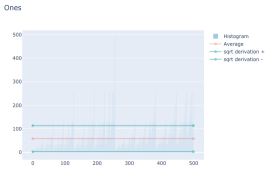
(b) 8 %



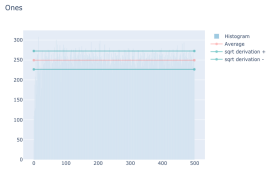
(c) 50 %



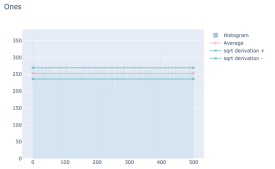
(d) 87 %



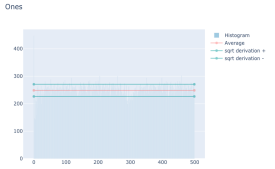
(a) One



(b) 8 %

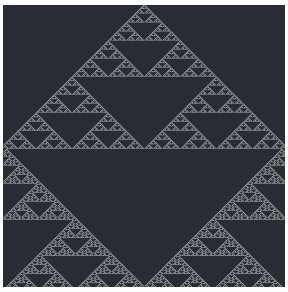


(c) 50 %

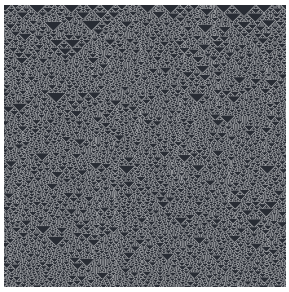


(d) 87 %

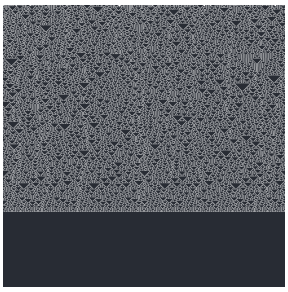
3.4.11. 146



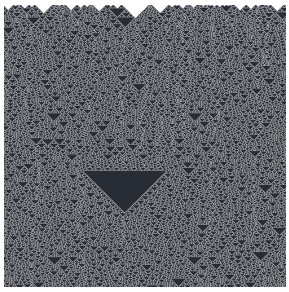
(a) One



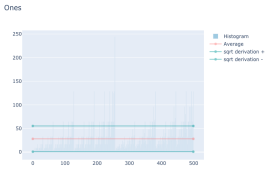
(b) 8 %



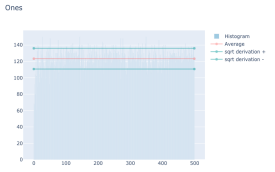
(c) 50 %



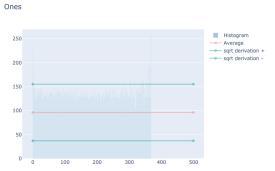
(d) 87 %



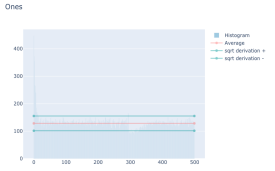
(a) One



(b) 8 %

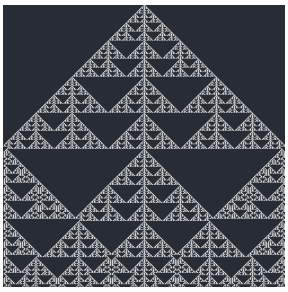


(c) 50 %

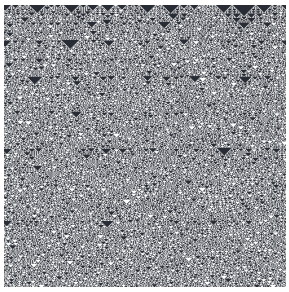


(d) 87 %

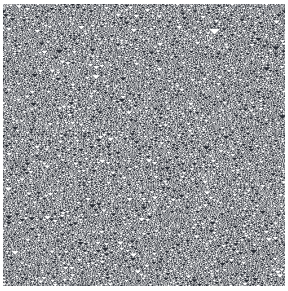
3.4.12. 150



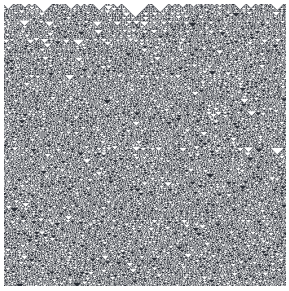
(a) One



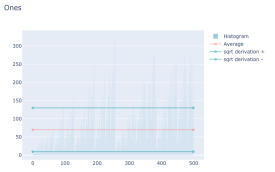
(b) 8 %



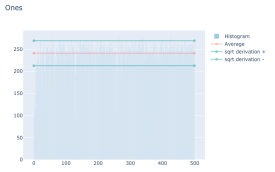
(c) 50 %



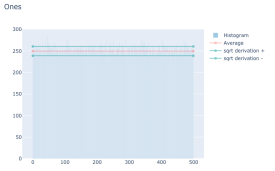
(d) 87 %



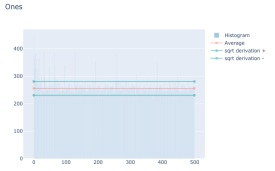
(a) One



(b) 8 %

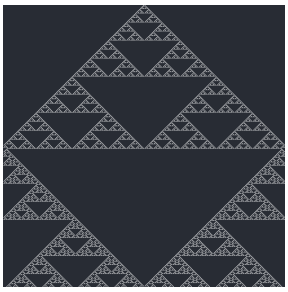


(c) 50 %

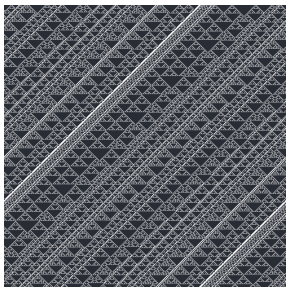


(d) 87 %

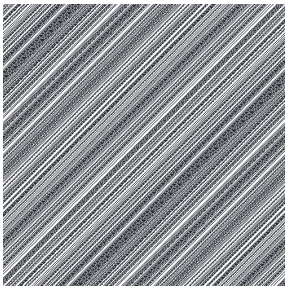
3.4.13. 154



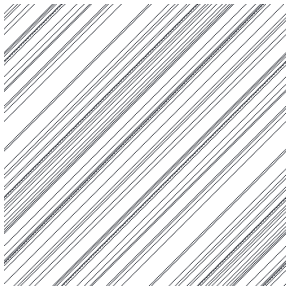
(a) One



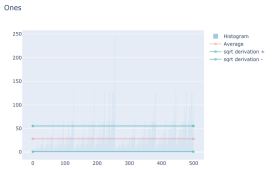
(b) 8 %



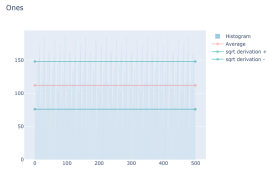
(c) 50 %



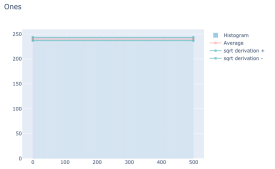
(d) 87 %



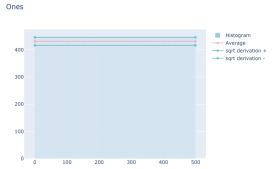
(a) One



(b) 8 %



(c) 50 %



(d) 87 %

3.5. W4

Surgen estructuras localizadas con interacciones complejas y localizadas veces de larga vida. Casi todos los patrones iniciales evolucionan en las estructuras que interactúan de manera compleja e interesante, con la formación de las estructuras locales que son capaces de sobrevivir por largos períodos de tiempo.

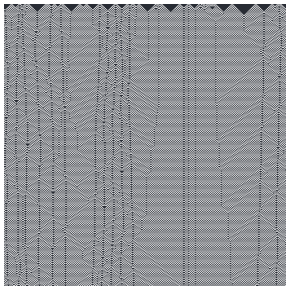
Podría ser el caso de que apareciesen estructuras estables u oscilantes, pero el número de pasos necesarios para llegar a este estado puede ser muy grande, incluso cuando el patrón inicial es relativamente simple.

Los cambios locales en el patrón inicial pueden extenderse indefinidamente.

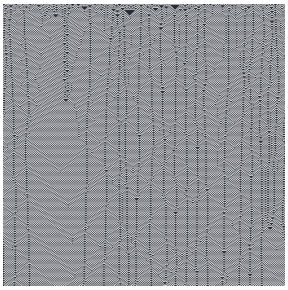
Puede exhibir cualquiera de los comportamientos anteriores simultáneamente y parecen poseer el tipo de complejidad que se encuentra en la intersección entre modelado matemático y estudios de vida.

3.5.1. 54

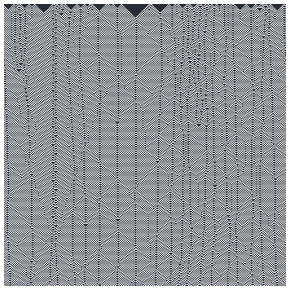
(a) One



(b) 8 %

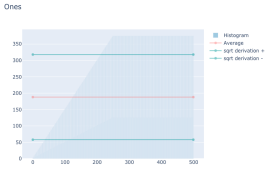


(c) 50 %

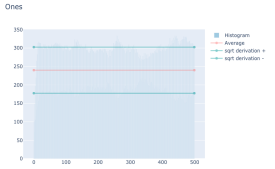


(d) 87 %

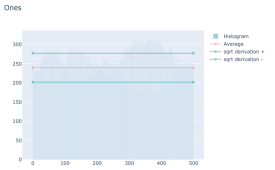
3.5.2. 73



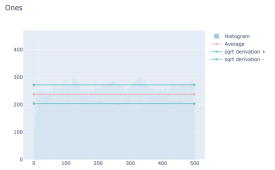
(a) One



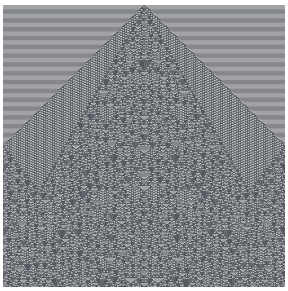
(b) 8 %



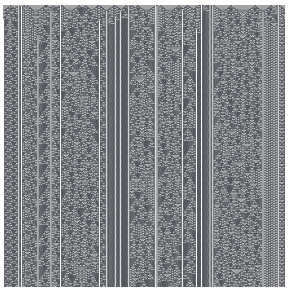
(c) 50 %



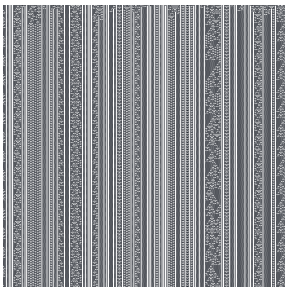
(d) 87 %



(a) One

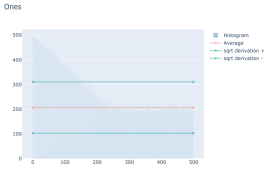


(b) 8 %

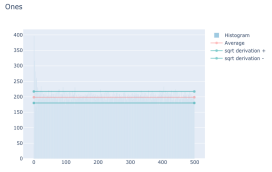


(c) 50 %

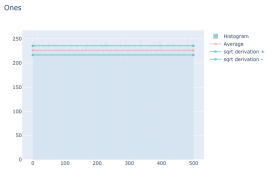
(d) 87 %



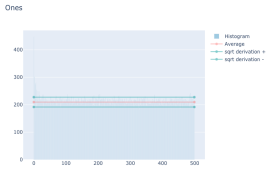
(a) One



(b) 8 %

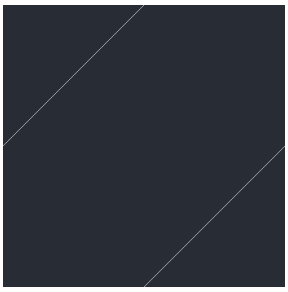


(c) 50 %

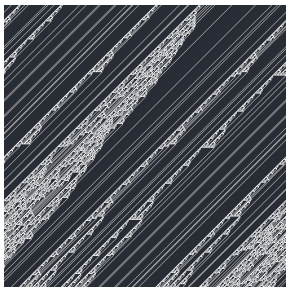


(d) 87 %

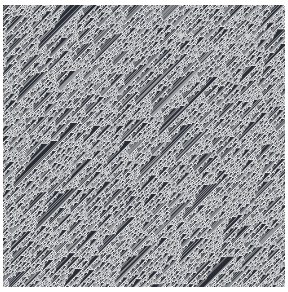
3.5.3. 106



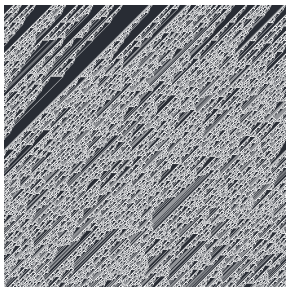
(a) One



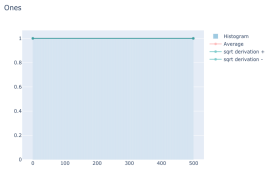
(b) 8 %



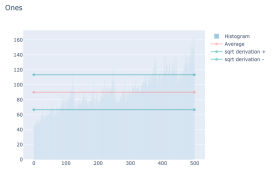
(c) 50 %



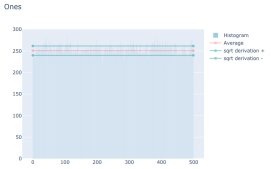
(d) 87 %



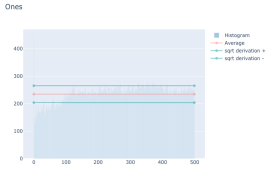
(a) One



(b) 8 %

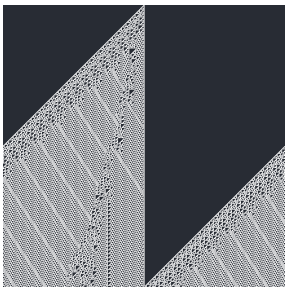


(c) 50 %

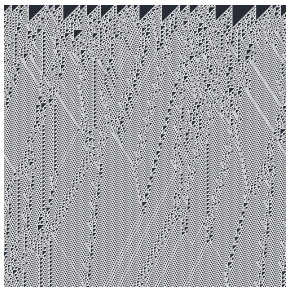


(d) 87 %

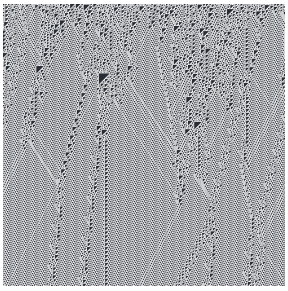
3.5.4. 110



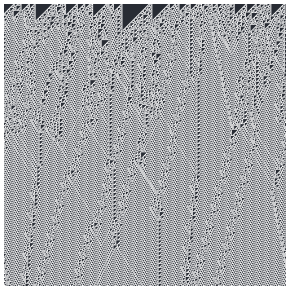
(a) One



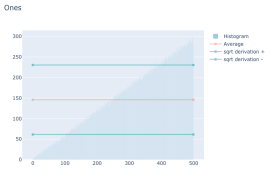
(b) 8 %



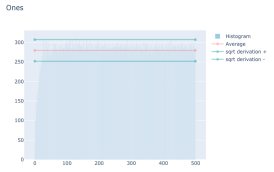
(c) 50 %



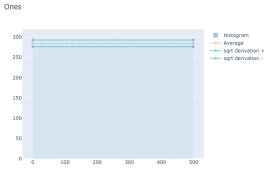
(d) 87 %



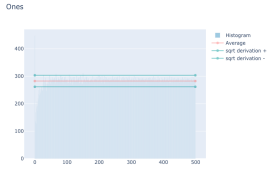
(a) One



(b) 8 %



(c) 50 %



(d) 87 %

Capítulo 4

Problemas de la regla 30

Del unico poblema en el que logre hacer algun avance es en el de intentar calcular el valor de la columna central.

Esta pregunta es uno de los tres desafíos planteados por Stephen Wolfram en el sitio web en https://rule30prize.org/.

Bibliografía

[1] *Cellular Automata*. Jarkko Kari, Spring 2013
`https://www.cs.tau.ac.il/~nachumd/models/CA.pdf`

[2] *A New Kind of Science*. Wolfram Stephen, 2002

[3] *The Structure of the Element ary Cellular Automata Rule Space*. Wentian Li, 1990
Santa Fe Institute, 1120 Canyon Road, Santa Fe, NM 87501, USA

[4] *Undecidability of CA Classification Schemes*. Karel C ulik II, 1988
Department of Compu ter Science, University of South Carolina, Columbia, SC 29208, USA

[5] *Introduccion a los automatas celulares elementales*. Carlos Zacarias Reyes Martinez
Escuela Superior de Computo