

VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS

STATISTINIS MODELIAVIMAS
Praktinės užduoties Nr. **1—11** ataskaita

Užduotį atliko: **Matas Gaulia**
Duomenų mokslas 2 kursas, 2 grupė

2021.05.12

Užduotis 1—11

1. Sugeneruokite pseudoatsitiktinių skaičių sekas tiesiniu kongruentiniu metodu su maksimaliu periodu, kai modulis $m = 1264$ ir $m = 729$.
Daugiklius a parinkite taip, kad galingumai b utų didžiausi. Prieauglio c parinkimui naudokitės gretimų narių koreliacija (teoriniai testai).
2. Gautas sekas patikrinkite su dviem testais. Pirma su intervalų testu.
Imkite intervalą $[3/4, 1]$. Kitą testą pasirinkite patys.
3. Naudodami sugeneruotą geresniją pseudoatsitiktinių skaičių seką sumodeliuokite du atsitiktinius dydžius, vieną pasiskirsčiusj pagal geometrinį skirstinį su parametru $p = 0.1$, o kitą parinkite patys.
4. Naudodami sugeneruotą geresniją pseudoatsitiktinių skaičių seką ir parinkdami tankius (tolygiai pasiskirsčiusio intervale $[0, 2]$ atsitiktinio dydžio ir kitų savo nuožiura) suskaičiuokite integralą:

$$\int_0^2 \frac{x + x^5}{1 + x} dx$$

Turinys

1 užduotis.....	4
Pirma seka:.....	4
Pirmos sekos kriterijai:.....	4
Skaičiaus a parinkimo kriterijai:	4
Skaičiaus c parinkimo kriterijai:	4
Pirmos sekos parametrai:	4
Pirmos sekos generavimas:.....	5
Antra seka:	6
Antra sekos kriterijai:	6
Skaičiaus a parinkimo kriterijai:	6
Skaičiaus c parinkimo kriterijai:	6
Antros sekos parametrai:.....	6
Antros sekos generavimas:	7
2 užduotis.....	8
Intervalų testas:	8
Skaitmenų testas:.....	8
3 užduotis.....	9
Geometrinis skirstinys:.....	9
Puasono skirstinys:.....	9
4 užduotis.....	10
Pirmas tankis:.....	10
Antras tankis:	10

1 užduotis

Sugeneruokite pseudoatsitiktinių skaičių sekas tiesiniu kongruentiniu metodu su maksimaliu periodu, kai modulis $m = 1264$ ir $m = 729$. Daugiklius a parinkite taip, kad galingumai būtų didžiausi. Prieauglio c parinkimui naudokitės gretimų narių koreliacija (teoriniai testai).

Pirma seka:

Pirma seka turi būti sugeneruota su $m = 1264$.

Pirmos sekos kriterijai:

Skaičiaus a parinkimo kriterijai:

- Turi būti mažesnis už 1264
- A-1 turi dalintis iš visų pirminių m daugiklių ($1264 = 2^4 \cdot 79$)
- A-1 turi dalintis iš 4 , jei m dalinasi iš 4. (Šiuo atveju tiesa, nes $1264/4 = 316$)

Skaičiaus c parinkimo kriterijai:

- $\text{DBD}(c,m) = \text{DBD}(c, 1264)$ turi būti lygu 1
- $c < m$
- $\frac{c}{m} \approx \frac{1}{2} \pm \frac{1}{6}\sqrt{3}$

Pirmos sekos parametrai:

Renkuosi a = 949:

- $949 < 1264$
- tada $a - 1 = 948$:
 - $948 / 2 = 474$
 - $948 / 79 = 12$
- $948 / 4 = 237$

Renkuosi $c = 631$:

- apskaičiuoju DBD(631, 1264) su jau parašyta Python funkcija gcd() iš bibliotekos "math":

```
In [2]: from math import gcd  
gcd(631,1264)  
Out[2]: 1
```

- $631 < 1264$
- $\frac{c}{m} = \frac{631}{1264} = 0.499$, labai arti 0.5, tad c parinktas tinkamai

Pirmą sekos narj ir pirmoje, ir antroje sekose imsiu 0.

Pirmajai sekai rasti tokie parametrai:

- $m = 1264$
- $a = 949$
- $c = 631$
- $X_0 = 0$

Pirmos sekos generavimas:

Generuoju seką su Python funkcija LCG (Linear Congruential Generator), funkcijos kodas parodytas priede:

```
In [4]: pirma = LCG(seed = 0, a = 949, c = 631, m = 1264)  
print(pirma)
```

[631, 314, 313, 628, 1259, 942, 941, 1256, 623, 306, 305, 620, 1251, 934, 933, 1248, 615, 298, 297, 612, 1243, 926, 925, 1240, 607, 290, 289, 604, 1239, 918, 917, 1232, 599, 282, 281, 596, 1227, 910, 909, 1224, 591, 274, 273, 588, 1219, 902, 901, 1216, 5 83, 266, 265, 580, 1211, 894, 893, 1208, 575, 258, 257, 572, 1203, 886, 885, 1208, 567, 250, 249, 564, 1195, 878, 877, 1192, 55 9, 242, 241, 556, 1187, 870, 869, 1184, 551, 234, 233, 548, 1179, 862, 861, 1176, 543, 226, 225, 540, 1171, 854, 853, 1168, 53 5, 218, 217, 532, 1163, 846, 845, 1160, 527, 210, 209, 524, 1155, 837, 837, 1152, 519, 202, 201, 516, 1147, 830, 829, 1144, 51 1, 194, 193, 508, 1139, 822, 821, 1136, 503, 186, 185, 500, 1131, 814, 813, 1128, 495, 178, 177, 492, 1123, 806, 805, 1120, 48 7, 170, 169, 484, 1115, 798, 797, 1112, 479, 162, 161, 476, 1107, 790, 789, 1104, 471, 154, 153, 468, 1099, 782, 781, 1096, 46 3, 146, 145, 460, 1091, 774, 773, 1088, 455, 138, 137, 452, 1083, 766, 765, 1080, 447, 130, 129, 444, 1075, 758, 757, 1072, 43 9, 122, 121, 436, 1067, 750, 749, 1064, 431, 114, 113, 428, 1059, 742, 741, 1056, 423, 106, 105, 420, 1051, 734, 733, 1048, 41 5, 98, 97, 412, 1043, 726, 725, 1040, 407, 90, 89, 404, 1035, 718, 717, 1032, 399, 82, 81, 396, 1027, 710, 709, 1024, 391, 74, 73, 388, 1019, 702, 701, 1016, 383, 66, 65, 380, 1011, 694, 693, 1008, 375, 58, 57, 372, 1003, 686, 685, 1000, 367, 58, 49, 36 4, 995, 678, 677, 992, 359, 42, 41, 356, 987, 670, 669, 984, 351, 34, 33, 348, 979, 682, 661, 976, 343, 26, 25, 340, 971, 654, 653, 968, 335, 18, 17, 332, 963, 646, 645, 960, 327, 10, 9, 324, 955, 638, 637, 952, 319, 2, 1, 316, 947, 630, 629, 944, 311, 1 258, 1257, 308, 939, 622, 621, 936, 303, 1250, 1249, 300, 931, 614, 613, 928, 295, 1242, 1241, 292, 923, 606, 605, 920, 287, 12 34, 1223, 284, 915, 598, 597, 912, 279, 1226, 1225, 276, 907, 590, 589, 904, 271, 1218, 1217, 268, 899, 582, 581, 896, 263, 12 0, 1209, 260, 891, 574, 573, 888, 255, 1202, 1201, 252, 883, 566, 565, 880, 247, 1194, 1193, 244, 875, 558, 557, 872, 239, 118 6, 1185, 236, 867, 550, 549, 864, 231, 1178, 1177, 228, 859, 542, 541, 856, 223, 1170, 1169, 228, 851, 534, 533, 848, 215, 116 2, 1161, 212, 843, 526, 525, 840, 207, 1154, 1153, 204, 835, 518, 517, 832, 199, 1146, 1145, 196, 827, 510, 509, 824, 191, 113 8, 1137, 188, 819, 502, 501, 816, 183, 1130, 1129, 180, 811, 494, 493, 808, 175, 1122, 1121, 172, 803, 486, 485, 800, 167, 111 4, 1113, 164, 795, 478, 477, 792, 159, 1106, 1105, 156, 787, 470, 469, 784, 151, 1098, 1097, 148, 779, 462, 461, 776, 143, 109 0, 1089, 140, 771, 454, 453, 768, 135, 1802, 1081, 132, 763, 446, 445, 750, 127, 1074, 1073, 124, 755, 438, 437, 752, 119, 106 6, 1065, 116, 747, 430, 429, 744, 111, 1058, 1057, 108, 739, 422, 421, 736, 1026, 1025, 76, 707, 390, 389, 704, 71, 1018, 1017, 6 8, 699, 382, 381, 696, 63, 1810, 1009, 60, 691, 374, 373, 688, 55, 1002, 1001, 52, 683, 366, 365, 680, 47, 994, 993, 44, 675, 3 58, 357, 672, 39, 986, 985, 36, 667, 350, 349, 664, 31, 978, 977, 28, 659, 342, 341, 656, 23, 970, 969, 20, 651, 334, 333, 648, 15, 962, 961, 12, 643, 326, 325, 640, 7, 954, 953, 4, 635, 318, 317, 632, 1263, 946, 945, 1260, 627, 310, 309, 624, 1255, 938, 937, 1252, 619, 302, 301, 616, 1247, 930, 929, 1244, 611, 294, 293, 608, 1239, 922, 921, 1236, 603, 286, 285, 600, 1231, 914, 9 13, 1228, 595, 278, 277, 592, 1223, 906, 905, 1220, 587, 270, 269, 584, 1215, 898, 897, 1212, 579, 262, 261, 576, 1207, 898, 88 9, 1204, 571, 254, 253, 568, 1199, 882, 881, 1196, 563, 246, 245, 560, 1191, 874, 873, 1188, 555, 238, 237, 552, 1183, 866, 86 5, 1180, 547, 230, 229, 544, 1175, 858, 857, 1172, 539, 222, 221, 536, 1167, 850, 849, 1164, 531, 214, 213, 528, 1159, 842, 84 1, 1156, 523, 206, 205, 520, 1151, 834, 833, 1148, 515, 198, 197, 512, 1143, 826, 825, 1140, 507, 190, 189, 504, 1135, 818, 81 7, 1132, 499, 182, 181, 496, 1127, 810, 809, 1124, 491, 174, 173, 488, 1119, 802, 801, 1116, 483, 166, 165, 480, 1111, 794, 79 3, 1108, 475, 158, 157, 472, 1103, 786, 785, 1100, 467, 158, 149, 464, 1095, 778, 777, 1092, 459, 142, 141, 456, 1087, 770, 76 9, 1084, 451, 134, 133, 448, 1079, 762, 761, 1076, 443, 126, 125, 440, 1071, 754, 753, 1068, 435, 118, 117, 432, 1063, 746, 74 5, 1060, 427, 110, 109, 424, 1055, 738, 737, 1052, 419, 102, 101, 416, 1047, 730, 729, 1044, 411, 94, 93, 408, 1039, 722, 721, 1036, 403, 86, 85, 400, 1031, 714, 713, 1028, 395, 78, 77, 392, 1023, 706, 705, 1020, 387, 70, 69, 384, 1015, 698, 697, 1012, 3 79, 62, 61, 376, 1007, 690, 689, 1004, 371, 54, 53, 368, 999, 682, 681, 996, 363, 46, 45, 360, 991, 674, 673, 988, 355, 38, 37, 352, 983, 666, 665, 980, 347, 30, 29, 344, 975, 658, 657, 972, 339, 22, 21, 336, 967, 650, 649, 964, 331, 14, 13, 328, 959, 64 2, 641, 956, 323, 6, 5, 320, 951, 634, 633, 948, 315, 1262, 1261, 312, 943, 626, 940, 387, 1254, 1253, 304, 935, 618, 617, 932, 299, 1246, 1245, 296, 927, 610, 609, 924, 291, 1238, 1237, 288, 919, 682, 681, 916, 283, 1230, 1229, 280, 911, 594, 593, 9 08, 275, 1222, 1221, 272, 903, 586, 585, 900, 267, 1214, 2123, 264, 895, 578, 577, 892, 259, 1206, 1205, 256, 887, 570, 569, 88 4, 251, 1198, 1197, 248, 879, 562, 561, 876, 243, 1190, 1189, 240, 871, 554, 553, 868, 235, 1182, 1181, 232, 863, 546, 545, 86 0, 227, 1174, 1173, 224, 855, 538, 537, 852, 219, 1166, 1165, 216, 847, 530, 529, 844, 211, 1158, 1157, 208, 839, 522, 521, 83 6, 203, 1150, 1149, 200, 831, 514, 513, 828, 195, 1142, 1141, 192, 823, 506, 505, 820, 187, 1134, 1133, 184, 815, 498, 497, 81 2, 179, 1126, 1125, 176, 807, 490, 489, 804, 171, 1118, 1117, 168, 799, 482, 481, 796, 163, 1110, 1109, 160, 791, 474, 473, 78 8, 155, 1102, 1101, 152, 783, 466, 465, 780, 147, 1094, 1093, 144, 775, 458, 457, 772, 139, 1086, 1085, 136, 767, 450, 449, 76 4, 131, 1078, 1077, 128, 759, 442, 441, 756, 123, 1078, 1069, 120, 751, 434, 433, 748, 115, 1062, 1061, 112, 743, 426, 425, 74 0, 107, 1054, 1053, 104, 735, 418, 417, 732, 99, 1046, 1045, 96, 727, 410, 409, 724, 91, 1038, 1037, 88, 719, 402, 401, 718, 8 3, 1030, 1029, 80, 711, 394, 393, 708, 75, 1022, 1021, 72, 703, 386, 385, 700, 67, 1014, 1013, 64, 695, 378, 377, 692, 59, 100 6, 1005, 56, 687, 370, 369, 684, 51, 998, 997, 48, 679, 362, 361, 676, 43, 990, 989, 40, 671, 354, 353, 668, 35, 982, 981, 32, 663, 346, 345, 660, 27, 974, 973, 24, 655, 338, 337, 652, 19, 966, 965, 16, 647, 330, 329, 644, 11, 958, 957, 8, 639, 322, 321, 636, 3, 950, 949, 0]

Antra seka:

Antra sekos kriterijai:

Skaičiaus a parinkimo kriterijai:

- Turi būti mažesnis už 729
- $a - 1$ turi dalintis iš visų pirminių 729 daugiklių ($729 = 3^6$)
- $a - 1$ turi dalintis iš 4 , jei 729 dalinasi iš 4. (Šiuo atveju netiesa, nes $729/4 = 182.25$)

Skaičiaus c parinkimo kriterijai:

- $\text{DBD}(c,m) = \text{DBD}(c, 729)$ turi būti lygu 1
- $c < m$
- $\frac{c}{m} \approx \frac{1}{2} \pm \frac{1}{6}\sqrt{3}$

Antros sekos parametrai:

Renkuosi a = 181:

- $181 < 729$
- $a - 1 = 180, 180/3 = 60$

Renkuosi c = 365:

- apskaičiuoju $\text{DBD}(365, 729)$ taip pat kaip ir pirmoje sekoje:

```
In [3]: from math import gcd  
gcd(365,729)  
Out[3]: 1
```

- $365 < 729$
- $\frac{c}{m} = \frac{365}{729} = 0.5007$, vel labai arti 0.5, tad c parinktas gerai.

Antrajai sekai rasti tokie parametrai:

- $m = 729$
- $a = 181$
- $c = 365$
- $X_0 = 0$

Antros sekos generavimas:

```
In [5]: antra = LCG(seed = 0, a = 181, c = 365, m = 729)
print(antra)

[365, 91, 69, 461, 700, 219, 638, 661, 450, 167, 703, 33, 506, 97, 426, 197, 301, 171, 698, 586, 726, 551, 223, 633, 485, 670,
621, 500, 469, 690, 596, 349, 111, 44, 310, 342, 302, 352, 654, 641, 475, 318, 332, 679, 63, 104, 235, 618, 686, 601, 525, 620,
319, 513, 635, 118, 582, 2, 727, 3, 179, 688, 234, 437, 1, 546, 47, 124, 210, 467, 328, 684, 239, 613, 510, 92, 250, 417, 26, 6
97, 405, 41, 496, 474, 137, 376, 624, 314, 337, 126, 572, 379, 438, 182, 502, 102, 602, 706, 576, 374, 262, 402, 227, 628, 309,
161, 346, 297, 176, 145, 366, 272, 25, 516, 449, 715, 18, 707, 28, 330, 317, 151, 723, 8, 355, 468, 509, 640, 294, 362, 277, 20
1, 296, 724, 189, 311, 523, 258, 407, 403, 408, 584, 364, 639, 113, 406, 222, 452, 529, 615, 143, 4, 360, 644, 289, 186, 497, 6
55, 93, 431, 373, 81, 446, 172, 150, 542, 52, 300, 719, 13, 531, 248, 55, 114, 587, 178, 507, 278, 382, 252, 50, 667, 78, 632,
304, 714, 566, 22, 702, 581, 550, 42, 677, 430, 192, 125, 391, 423, 383, 433, 6, 722, 556, 399, 413, 31, 144, 185, 316, 699, 3
8, 682, 606, 701, 400, 594, 716, 199, 663, 83, 79, 84, 260, 40, 315, 518, 82, 627, 128, 205, 291, 548, 409, 36, 320, 694, 591,
173, 331, 498, 107, 49, 486, 122, 577, 555, 218, 457, 705, 395, 418, 207, 653, 460, 519, 263, 583, 183, 683, 58, 657, 455, 343,
483, 308, 709, 390, 242, 427, 378, 257, 226, 447, 353, 106, 597, 530, 67, 99, 59, 109, 411, 398, 232, 75, 89, 436, 549, 590, 72
1, 375, 443, 358, 282, 377, 76, 270, 392, 604, 339, 488, 484, 489, 665, 445, 720, 194, 487, 303, 533, 610, 696, 224, 85, 441, 7
25, 370, 267, 578, 7, 174, 512, 454, 162, 527, 253, 231, 623, 133, 381, 71, 94, 612, 329, 136, 195, 668, 259, 588, 359, 463, 33
3, 131, 19, 159, 713, 385, 66, 647, 103, 54, 662, 631, 123, 29, 511, 273, 206, 472, 504, 464, 514, 87, 74, 637, 480, 494, 112,
225, 266, 397, 51, 119, 34, 687, 53, 481, 675, 68, 280, 15, 164, 160, 165, 341, 121, 396, 599, 163, 708, 209, 286, 372, 629, 49
0, 117, 401, 46, 672, 254, 412, 579, 188, 130, 567, 203, 658, 636, 299, 538, 57, 476, 499, 288, 5, 541, 600, 344, 664, 264, 35
139, 9, 536, 424, 564, 389, 61, 471, 323, 508, 459, 338, 307, 528, 434, 187, 678, 611, 148, 180, 140, 190, 492, 479, 313, 156,
170, 517, 630, 671, 73, 456, 524, 439, 363, 458, 157, 351, 473, 685, 420, 569, 565, 570, 17, 526, 72, 275, 568, 384, 614, 691,
48, 305, 166, 522, 77, 451, 348, 659, 88, 255, 593, 535, 243, 608, 334, 312, 704, 214, 462, 152, 175, 693, 410, 217, 276, 20, 3
40, 669, 440, 544, 414, 212, 100, 240, 65, 466, 147, 728, 184, 135, 14, 712, 204, 110, 592, 354, 287, 553, 585, 545, 595, 168,
155, 718, 561, 575, 193, 306, 347, 478, 132, 200, 115, 39, 134, 562, 27, 149, 361, 96, 245, 241, 246, 422, 202, 477, 680, 244,
68, 290, 367, 453, 710, 571, 198, 482, 127, 24, 335, 493, 660, 269, 211, 648, 284, 10, 717, 388, 619, 138, 557, 580, 369, 86, 6
22, 681, 425, 16, 345, 116, 220, 90, 617, 505, 645, 470, 142, 552, 404, 589, 540, 419, 388, 609, 515, 268, 30, 692, 229, 261, 2
21, 271, 573, 560, 394, 237, 251, 598, 711, 23, 154, 537, 605, 520, 444, 539, 238, 432, 554, 37, 501, 650, 646, 651, 98, 607, 1
53, 356, 649, 465, 695, 43, 129, 386, 247, 603, 158, 532, 429, 11, 169, 336, 674, 616, 324, 689, 415, 393, 56, 295, 543, 233, 2
56, 45, 491, 298, 357, 101, 421, 21, 521, 625, 495, 293, 181, 321, 146, 547, 228, 80, 265, 216, 95, 64, 285, 191, 673, 435, 36
8, 634, 666, 626, 676, 249, 236, 70, 642, 656, 274, 387, 428, 559, 213, 281, 196, 120, 215, 643, 108, 230, 442, 177, 326, 322,
327, 503, 283, 558, 32, 325, 141, 371, 448, 534, 62, 652, 279, 563, 208, 105, 416, 574, 12, 350, 292, 0]
```

2 užduotis

Gautas sekas patikrinkite su dviem testais. Pirma su intervalų testu.
Imkite intervalą $[3/4, 1)$. Kitą testą pasirinkite patys.

Intervalų testas:

Kad patestuočiau abi sekas su interval testu, pasirašiau tam funkciją, kuri priima 3 būtinus argumentus: seką sąrašo pavidalu, m skaičių, t skaičių.

Abiejoms sekoms testuoti rinkausi $t = 4$.

```
In [6]: print("Pirma seka:")
intervalu_testas(pirma,m = 1264, t = 4)

print("Antra seka:")
intervalu_testas(antra,m = 729, t = 4)

Pirma seka:
Intervalų testas: 230.09
Antra seka:
Intervalų testas: 0.46
```

Labai gerai matosi, kad antroji seka yra žymiai geresnė pagal intervalų testą.

Skaitmenų testas:

Skaitmenų testui taip pat pasirašiau funkciją, kuri priima vieną argumentą – seką sąrašo pavidalu.

```
In [7]: print("Pirma seka:")
skaitmenu_testas(pirma)

print("Antra seka:")
skaitmenu_testas(antra)

Pirma seka:
Skaitmenu testas: 311.63
Antra seka:
Skaitmenu testas: 110.55
```

Iš skaitmenų testo matyti taip pat, kad antroji seka yra žymiai geresnė už pirmąją, tad ją naudosiu 3 ir 4 užduotyse.

3 užduotis

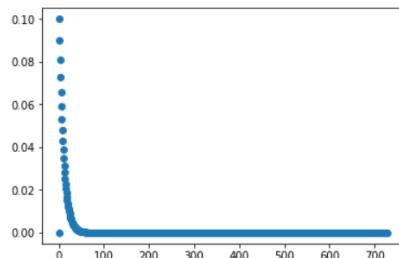
Naudodami sugeneruotą geresniją pseudoatsitiktinių skaičių seką sumodeliuokite du atsitiktinius dydžius, vieną pasiskirsčiusį pagal geometrinj skirstinj su parametru $p = 0.1$, o kitą parinkite patys.

Geometrinis skirstinys:

Kad sugeneruočiau geometrinj a.d., naudojau Python biblioteką “scipy”, funkciją geom.pmf()

```
In [8]: from scipy.stats import geom  
geom_pd = geom.pmf(antra, p = 0.1)  
plt.plot(antra, geom_pd, 'o')
```

```
Out[8]: [<matplotlib.lines.Line2D at 0x759390>]
```

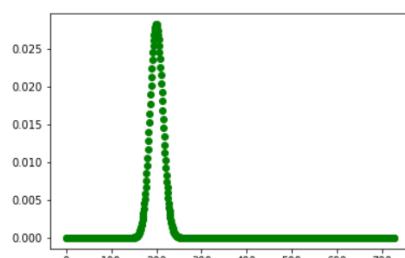


Puasono skirstinys:

Kitą a.d. pasirinkau puasono su $\lambda = 200$, vėl naudojau biblioteką “scipy”, funkciją poisson.pmf()

```
In [9]: from scipy.stats import poisson  
pos = poisson.pmf(antra, 200)  
plt.plot(antra, pos, 'o', color = "green")
```

```
Out[9]: [<matplotlib.lines.Line2D at 0x19dc1d0>]
```



4 užduotis

Naudodami sugeneruotį geresnįją pseudoatsitiktiniu skaičiu seką ir parinkdami tankius (tolygiai pasiskirsčiusio intervale $[0, 2]$ atsitiktinio dydžio ir kitų savo nuožiura) suskaičiuokite integralą:

$$I = \int_0^2 \frac{x + x^5}{1 + x} dx$$

Generuosiu tolygū atsitiktinj dydi tarp $[0,1]$ su python paketu “numpy”, funkcija `numpy.random.uniform(0,1,N)`

Kad būtų galima atkartoti atsakymus, nustačiau atsitiktinių skaičių seką su `np.random.seed(69420)`

```
In [10]: np.random.seed(69420)
N = len(antra)
uni = np.random.uniform(0,1,N)
```

Pirmas tankis:

Pirmas tankis bus $p_{\varepsilon 1}(x) = \frac{1}{2}$; $\int_0^2 \frac{1}{2} dx = 1$, tada $\int_0^{\varepsilon} \frac{1}{2} dx = \frac{\varepsilon}{2}$
 $\frac{\varepsilon}{2} = U$, tai $\varepsilon = 2U$

Galime skaičiuoti integralą $I \approx \frac{1}{N} \sum_{j=1}^N \left(\frac{\varepsilon_j + \varepsilon_j^5}{1 + \varepsilon_j} \right) \cdot \frac{2}{1} = \frac{2}{N} \sum_{j=1}^N \frac{\varepsilon_j + \varepsilon_j^5}{1 + \varepsilon_j}$

```
In [11]: suma_1 = 0
epsilon_1 = [2*u for u in uni]
for i in epsilon_1:
    suma_1+=(2/N)*(i+i**5)/(1+i)
print(suma_1)
```

4.612236451187102

Su pirmiu tankiu gauname, kad integralas lygus apytiksliai **4.61**

Antras tankis:

Antras tankis bus $p_{\varepsilon 2}(x) = \frac{x^3}{4}$; $\int_0^2 \frac{x^3}{4} dx = 1$, tada $\int_0^{\varepsilon} \frac{x^3}{4} dx = \frac{\varepsilon^4}{16}$
 $\frac{\varepsilon^4}{16} = U$, tai $\varepsilon = 2\sqrt[4]{U}$

Galime skaičiuoti integralą $I \approx \frac{1}{N} \sum_{j=1}^N \left(\frac{\varepsilon_j + \varepsilon_j^5}{1 + \varepsilon_j} \right) \cdot \frac{4}{\varepsilon_j^3} = \frac{4}{N} \sum_{j=1}^N \frac{\varepsilon_j + \varepsilon_j^5}{\varepsilon_j^3(1 + \varepsilon_j)}$

```
In [12]: suma_2 = 0
epsilon_2 = [2*(u**(1/4)) for u in uni]
for i in epsilon_2:
    suma_2+=(4/N)*(i+i**5)/((1+i)**3)
print(suma_2)
```

4.876908512535559

Su antru tankiu gaume, kad integralas lygus apytiksliai **4.88**

Geresnis tankis yra antrasis, nes artimesnis

Išvada

Antra sugeneruota seka yra geresnė pagal intervalų ir skaitmenų testus.

Skaičiuojant duotą integralą 4 užduotyje, nustačiau, kad geresnis rezultatas gaunamas naudojant antrą tankį.

Priedas

```
from collections import Counter
from sortedcontainers import SortedDict
import numpy as np
import matplotlib.pyplot as plt

def uzpildyti_nulius(counts):
    app = {}
    for idx in range(min(counts), max(counts) + 1):
        val = counts.get(idx)
        if val == None:
            val = 0
        app[idx] = val
    return SortedDict(dict(app))

def LCG(seed, a, c, m):
    numbers = []
    for i in range(m):
        seed = (a * seed + c) % m
        numbers.append(seed)

    return numbers

def intervalu_testas(seq, m, a = 0.75, b = 1, t = 4):
    seq = [i/m for i in seq]
    intervals = []
    temp = []
    for num in seq:
        if num >= a and num <=b:
            temp.append(num)
            intervals.append(temp)
            temp = []
        else:
            temp.append(num)

    interval_lengths = [len(i)-1 for i in intervals]
    counts = SortedDict(dict(Counter(interval_lengths)))
    counts = uzpildyti_nulius(counts)
    n = sum(counts.values())
    p = b-a
    chisq = 0
    for key, value in counts.items():

        if key <= t:
            chisq += ((value - n * (p * (1-p)**key)) )**2) / (n * (p * (1-p)**key))
        else:
            abovet = 0
            for key in counts.keys():
                if key > t:
                    abovet += counts[key]
            chisq += ((abovet - n * ((1-p)**(t+1))) )**2) / (n * ((1-p)**(t+1)))
            break
```

```

print("Intervalu testas: ", round(chisq,2))

def skaitmenu_testas(seka):
    counts = {0:0,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0}
    for i in [str(num) for num in seka]:
        for char in i:
            counts[int(char)]+=1
    suma = sum(counts.values())
    result = 0
    for val in counts.values():
        result += ((val - suma/10)**2)/(suma/10)
    print("Skaitmenu testas:", round(result,2))

# In[2]:


from math import gcd
gcd(631,1264)

# In[3]:


from math import gcd
gcd(365,729)

# In[4]:


pirma = LCG(seed = 0, a = 949, c = 631, m = 1264)
print(pirma)

# In[5]:


antra = LCG(seed = 0, a = 181, c = 365, m = 729)
print(antra)

# In[6]:


print("Pirma seka:")
intervalu_testas(pirma,m = 1264, t = 4)

print("Antra seka:")
intervalu_testas(antra,m = 729, t = 4)

# In[7]:


print("Pirma seka:")
skaitmenu_testas(pirma)

print("Antra seka:")
skaitmenu_testas(antra)

```

```
# In[8]:
```

```
from scipy.stats import geom

geom_pd = geom.pmf(antra, p = 0.1)
plt.plot(antra, geom_pd, 'o')
plt.show()
```

```
# In[9]:
```

```
from scipy.stats import poisson
pos = poisson.pmf(antra, 200)
plt.plot(antra, pos, 'o', color = "green")
plt.show()
```

```
# In[10]:
```

```
np.random.seed(69420)
N = len(antra)
uni = np.random.uniform(0,1,N)
```

```
# In[11]:
```

```
suma_1 = 0
epsilon_1 = [2*u for u in uni]
for i in epsilon_1:
    suma_1+=(2/N)*(i+i**5)/(1+i)
print(suma_1)
```

```
# In[12]:
```

```
suma_2 = 0
epsilon_2 = [2*(u**(1/4)) for u in uni]
for i in epsilon_2:
    suma_2+=(4/N)*(i+i**5)/((1+i)*(i**3))
print(suma_2)
```