# 1.1: Problem Statement:

Although the sequence of pseudo-random numbers is not truly uniform, a good generator will produce numbers that gives essentially the same results as true random numbers. Generation of random numbers has numerous applications in many scientific problems. The problem asks to generate random numbers from a uniform random generator and use Kolmogorov-Smirnov test to find if the samples come from a uniform distribution. For the random number generation process we are going to use multiplicative congruential method.

# 1.2: Test Description:

The Kolmogorov- Smirnov test compares the sample distribution of a set of N generated random numbers with the distribution function , of a uniformly distributed random variable. Here is defined as,

Let α be the significance level of the test, i.e. α is the probability of rejecting the null hypothesis that the numbers are uniformly distributed on the interval given that the null hypothesis is true. Under the null hypothesis will tend to theoretical probability as N tends to infinity. The test statistic is

Where D is the maximum absolute difference between and over the range of the random variable. If the value D is greater than some critical value , the null hypothesis is rejected. If D, we conclude that no difference has been detected between the sample distribution of the generated numbers and the uniform distribution and we don’t reject null hypothesis. The critical value is given by,

Where n is the sample size.

# Uniform Random Number Generation

Multiplicative congruential number generators are of the form . We considered the following algorithm for random number generation:

Where is the initial value or seed.

Random number generator should have sufficiently long period so that they don’t repeat in the simulation. Repeated periods can severely compromise the simulation results. Using a long period () should ensure this property.

Here is a frequency distribution of our uniform random number generator for n=10000

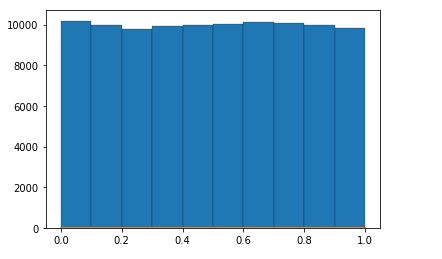


Fig: Histogram of the proposed uniform random number generator for n=100000

Now by visualizing at the plot we can see that the distribution of our random numbers looks like uniform. Now what happens if we increase the size of n?

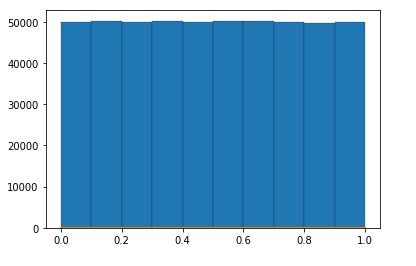


Fig: Histogram of the proposed random number generator for n=500000

We can see that as n is increased, the frequency of random number generator takes a flatter distribution. So that as n increases, numbers from our uniform random number generator has almost equal probability to be selected which is a property of uniform random numbers. But to conclude if our uniform random generator has a uniform distribution, we need to do some tests.

# Test Results

One of the tests we are going to do is called Kolmogorov-Smirnov which tests if a sample distribution follows a theoretical distribution. Using initial seed in python gives us a value of 1510 as . Our test passed successfully.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Initial Value | Test Statistic | Critical Value | p-value | Test Statistic<Critical Value? |
| 1510 | 0.027028492291005568 | 0.0514692312 | 0.45572533227891227 | Yes |

Here is the output:

KstestResult(statistic=0.027028492291005568, pvalue=0.45572533227891227)

[0.011817817581732672, 0.6220600961810258, 0.964036514500173, 0.5616992044084236, 0.4785284923755231, 0.6283713554164261, 0.037370483874050196, 0.08572247116161626, 0.7375728132843845, 0.3862728706497107, 0.08813700968778553, 0.31872182261139237, 0.7576726296719501, 0.20388689646678368, 0.7270689172330633, 0.8472919360954743, 0.43556995663585607, 0.6242611788326228, 0.9576326398912969, 0.9317786530273867, 0.4038214312884125, 0.026795664349010057, 0.3547307138120433, 0.959107039011599, 0.7120046679452083, 0.6624541551165535, 0.866985043914516, 0.41763307127060045, 0.1590288449819334, 0.797797611354756, 0.5844540393838911, 0.919039925056994, 0.3040204328969216, 0.6714156985615454, 0.48364572389221083, 0.633681456387826, 0.28423751019138727, 0.17983378664582678, 0.4664521564107631, 0.6613927956956405, 0.02871725663017354, 0.650932183326656, 0.21720517110880705, 0.5673108257201085, 0.7930478778635375, 0.7556832524741456, 0.7684243329653629, 0.9077641488554721, 0.7920498139187926, 0.9812225331464888, 0.4071145930360605, 0.37496515706878397, 0.03939485505195095, 0.10932885813961218, 0.4901187524619134, 0.4258726273783821, 0.6412483484676333, 0.4609926955127123, 0.9042334821560576, 0.45213459686009894, 0.026169427682724514, 0.8295710635509207, 0.6008651003245568, 0.7397411548251943, 0.8295891470413604, 0.9047943241450909, 0.8782059065430453, 0.006671268961704927, 0.12401743937470831, 0.3611035707225574, 0.067713134022296, 0.054643512728923704, 0.3935184350207068, 0.8643373930195055, 0.9185644788288346, 0.3131956762230004, 0.8797302799670632, 0.6268154064318237, 0.8865358996607996, 0.008865599058971553, 0.004123384134901401, 0.3017171552878419, 0.960228922758358, 0.5675047997233946, 0.053168951092832234, 0.6105610172313457, 0.699016607226346, 0.37211765319673235, 0.18139727748064197, 0.7440426171496709, 0.12426643451874445, 0.5459649565378041, 0.033024530873179685, 0.043290385530931126, 0.5815096183593895, 0.4321557662599514, 0.24196353100331663, 0.6810655727428689, 0.6690810893983957, 0.24586951883783076, 0.32900310742156724, 0.5552264342807356, 0.6906809563239482, 0.27483293659744457, 0.11716539325060574, 0.19876436293067615, 0.6326477758738435, 0.9111691116873031, 0.0192601285033208, 0.7049797553126605, 0.5947475398866215, 0.9219028744482914, 0.42161085243411867, 0.013596860232575265, 0.5224299288924923, 0.47981489611780964, 0.24895905202671842, 0.2547874130563752, 0.21205123849774302, 0.9451654315670791, 0.39540834789881874, 0.6281031354461345, 0.5293974431834172, 0.5828275836924219, 0.5831991185355927, 0.8275852277072078, 0.22492207504106781, 0.26531521522687523, 0.15282231809237148, 0.4847001784875524, 0.3558998402934055, 0.6086158112662918, 0.00593995256626045, 0.8327827811393806, 0.5802026095707913, 0.4652590562893353, 0.608959054857939, 0.7748349973814725, 0.6518009904081938, 0.8192457905128812, 0.06400114999339038, 0.6673279389121234, 0.7806692960582065, 0.7088588502764976, 0.790696597094972, 0.2377073751938098, 0.14785488236129976, 0.9970078463652208, 0.7108738602655352, 0.6569694828507349, 0.6860982723003711, 0.2536625523370051, 0.30651712804404885, 0.6333710363290138, 0.06700758173456768, 0.19642621287909626, 0.335359858970791, 0.3931497220849384, 0.6673790815600097, 0.6402237790824025, 0.24105503793854965, 0.41202263320424715, 0.8643962637821195, 0.9080053860824581, 0.8465238878720086, 0.5269834648477768, 0.011093696584503026, 0.45175849574234267, 0.7050379415531819, 0.5726836843288893, 0.09468251564292354, 0.32904041061598827, 0.1821812229148025, 0.9198135290852811, 0.3059833363192078, 0.6619335169261011, 0.11661897698259865, 0.015146146535475807, 0.5612848217418812, 0.5139990157978604, 0.781458514640787, 0.9732555677058434, 0.5063264321099624, 0.8283444721383716, 0.9855432296104465, 0.025060062774019346, 0.1844750429431326, 0.4720467452295342, 0.6896470727816443, 0.8983522410962508, 0.6061161046876181, 0.9933714847980866, 0.5945450014409353, 0.5178392177996408, 0.3237335585633915, 0.9899187749204779, 0.5648500884719426, 0.43543694793965526, 0.38878402178584787, 0.2930541547448626, 0.3611787969065731, 0.33203960877472516, 0.5897046768058579, 0.16650307605345877, 0.4171992304814976, 0.8674667025299122, 0.5128694202345188, 0.7963458815572532, 0.185231332753427, 0.18300958684785737, 0.8421261519389814, 0.6142356384611855, 0.4583756171438729, 0.9189973370726208, 0.5882441795376335, 0.6199254890065293, 0.08769373273835225, 0.8685661334863706, 0.9910055054309803, 0.8295297784868301, 0.9069870281531415, 0.7309821698493241, 0.6173286575904715, 0.442748123054741, 0.2677041810321175, 0.30417060679950314, 0.19538847924926714, 0.8941707424326664, 0.32766806582346003, 0.11718229489269773, 0.4828302615707881, 0.9282062202357716, 0.361943502613317, 0.1844484220186474, 0.0246288674066909, 0.9373745042539083, 0.45329299543671914, 0.49537430493877005, 0.7559431059080842, 0.13578099717189604, 0.07121946805679215, 0.9855996305055914, 0.9729899074756493, 0.04137494323839198, 0.3886710076540108, 0.3936256409593046, 0.666147603032248, 0.9427641629906204, 0.03728738335766242, 0.6890520922323, 0.898514148266294, 0.3272899116050871, 0.7615443466983476, 0.2758349591287947, 0.9581580776526397, 0.7628111079162039, 0.5662907476379959, 0.6485955517965348, 0.9454390443607416, 0.9940185709828597, 0.47012250892358015, 0.3490074786120129, 0.7686930321011194, 0.42379052351405405, 0.6473287007060501, 0.6534727665844712, 0.91678798520788, 0.45566738883762964, 0.40180419404143664, 0.12308925442541449, 0.7610991279413454, 0.7930433101919682, 0.6789143964084398, 0.5142604366476929, 0.1751587377745466, 0.8929057768047348, 0.06739075717860403, 0.6364559007978328, 0.9143247091743745, 0.055387093711359005, 0.8908840068107862, 0.08750246888375957, 0.6539945293469329, 0.6860547339013101, 0.5219126793192292, 0.7864013182867324, 0.0469564451123385, 0.19697300307311724, 0.5252626498813101, 0.08935655517939317, 0.8156229000611337, 0.17408132747471394, 0.784870867517251, 0.32467036243745606, 0.7347814863243985, 0.472440654166248, 0.3100745721301411, 0.4233337912817177, 0.9710300718299253, 0.10241724555493205, 0.32664604174282685, 0.9400235716905555, 0.9761694031656577, 0.47915900520941196, 0.2254005545868541, 0.3071209412566949, 0.7816597012717554, 0.35459927439438144, 0.7500047463690884, 0.3297722252690104, 0.4817900962577155, 0.4461478034249264, 0.4061321627377217, 0.8632591328878231, 0.796246445642899, 0.5140119202034603, 0.9983428595579894, 0.1484405911287482, 0.8410151008707542, 0.9408003347650172, 0.031226395643887295, 0.8220315868137551, 0.8848795787826551, 0.1710806000843088, 0.35164561697824187, 0.10788455331133891, 0.21568750367299072, 0.05987423195497796, 0.3062164673145006, 0.5801661548112361, 0.8525639124459419, 0.04167647894549951, 0.4565816370102491, 0.7675732312572995, 0.6032977414332785, 0.6251402691123729, 0.7325029716512668, 0.17744454284079583, 0.31043152525575435, 0.4226449734636792, 0.39406900405607603, 0.11775117047026343, 0.04392209371734508, 0.19862910741876302, 0.35940838715033996, 0.5767628357637501, 0.652980681347186, 0.6463114021561627, 0.5557360386269801, 0.2556012036537757, 0.8894298090084595, 0.6468000051783398, 0.7676870323567125, 0.5159528192672659, 0.6190334249376475, 0.09477292704152546, 0.8485847869182866, 0.1645137356428959, 0.9823549501515715, 0.43964719746245406, 0.15044775146546202, 0.5753588800203795, 0.05669650251823315, 0.8981178239444819, 0.6662670349079496, 0.9500556979095822, 0.5861147663491382, 0.8308780299643418, 0.5670496106925651, 0.4028069099424439, 0.9757354026547332, 0.1849124181014078, 0.8230110303605958, 0.3463872705336601, 0.7308558592250831, 0.49442599597127457, 0.817714289211535, 0.3240587782692438, 0.45588637118036224, 0.08224042834818383, 0.21487924792565372, 0.4755198864617943, 0.06273176337719512, 0.33274708051828067, 0.4801822707430377, 0.42342437823462503, 0.49352498934302713, 0.6744958882566988, 0.2523939303366439, 0.984787167974183, 0.31793214209281473, 0.48551215393725417, 0.0027712234308809153, 0.5759522028155403, 0.028672720784634686, 0.9024182273551906, 0.9431471586893998, 0.4742960927422606, 0.4944307191737139, 0.897097152609889, 0.5118439144044388, 0.5606693954023856, 0.17052852789430345, 0.07296831955805809, 0.3785468122821985, 0.23627402690997068, 0.057570275877402294, 0.5836266715003302, 0.013467906049204947, 0.3550969689875361, 0.1147577735198465, 0.733899548060214, 0.6497042480156311, 0.5792963987119945, 0.2345731524911584, 0.470973918899416, 0.6586549424839462, 0.013618327683591436, 0.8832333781212723, 0.5033860842247428, 0.40991756525352485, 0.48451921599196235, 0.31446317691098113, 0.18261434286023226, 0.19926045192371145, 0.9704154818181021, 0.7730029168413034, 0.860023351786669, 0.41247347854658656, 0.44175393248058575, 0.5583432012043629, 0.07418264172700356, 0.7876595057489627, 0.1933131228169953, 0.013655185240160295, 0.5026983313740689, 0.8508554039759819, 0.3267746243284897, 0.10111108892648997, 0.37407158751695957, 0.021171397539401147, 0.827678444715067, 0.7916203261314054, 0.7628212905315781, 0.7374299642338557, 0.9854088784127537, 0.7670194831523204, 0.29645334104842197, 0.49130300082792666, 0.3295349149636621, 0.4933157942692357, 0.15855428304455907, 0.8218351299044839, 0.583028304662103, 0.9567164559647052, 0.5334753988000915, 0.12102763313847949, 0.11143015842485715, 0.8066726465740579, 0.747170970191793, 0.7024960134655684, 0.8504983158085953, 0.3251937950612948, 0.532113595182129, 0.2331942260419923, 0.29535708776458963, 0.06657405945778548, 0.9102173070005222, 0.022278757776263522, 0.4390819456610279, 0.6502607248957551, 0.9320033229570851, 0.17984893972978414, 0.721130038481732, 0.03255676246832905, 0.18150680520641935, 0.5848751042899094, 0.995877800507414, 0.7181931281081368, 0.6719041134565622, 0.6924348644411353, 0.7527666621621543, 0.7492909593271515, 0.3331534114355004, 0.3093859964559721, 0.8504424355227698, 0.3860138311917027, 0.7344608389467284, 0.08332017766466372, 0.3622260100032324, 0.932550124326977, 0.36993956350252943, 0.5742437870121765, 0.3153283136502506, 0.7229675197615137, 0.915104631760672, 0.16354600161479135, 0.717649139798083, 0.529092586380007, 0.45909928877796014, 0.08174649117595818, 0.9132771943292008, 0.4498050908789994, 0.8741624033423897, 0.047512975543510624, 0.5505799597830419, 0.5973840735840537, 0.23412472719053026, 0.9342898912421846, 0.6102021073970022, 0.6668190214162781, 0.22729294338603176, 0.11249948903568996, 0.7789122228412481, 0.17772929285547198, 0.09622502191747773, 0.2539433670481403, 0.02616997809436637, 0.8388218320155618, 0.07853068554705507, 0.8652319893544689, 0.9540450805584179, 0.6356689453291097, 0.6879641463458371, 0.6134076344843058, 0.5421127777277086, 0.28945526959814843, 0.8747161360805463, 0.3540991057428062, 0.34367021934300207, 0.06537649783556186, 0.7827991222882639, 0.5048482988517956, 0.9853588021292159, 0.925387385732209, 0.9857920012370646, 0.20616479134474172, 0.011648131074220002, 0.7701389644155926, 0.7255749328646692, 0.7378966564954709, 0.8291057193787329, 0.7798255983646147, 0.5288317140791713, 0.07461852863180382, 0.11361071472690008, 0.4552824150097009, 0.9315490680427985, 0.5451865953138035, 0.9511074390966014, 0.2627288965800446, 0.6845648208095528, 0.480943346154384, 0.2148188167320652, 0.45985281581983567, 0.7462754839781092, 0.6520592200812229, 0.1593119051117971, 0.5551892139740238, 0.06511926141805913, 0.4594266533197028, 0.5837623442447569, 0.29371972162915383, 0.5473614211880422, 0.5034059074257528, 0.7430861046272731, 0.048160470578894235, 0.4330290194754624, 0.9187303240963864, 0.10055708796743168, 0.06297746862423488, 0.4623151675156854, 0.13102043612442, 0.0604699431268824, 0.31833413351249606, 0.2417819445216013, 0.6291415745528143, 0.9824435091495717, 0.9280582768507574, 0.8754590306782438, 0.8399286092444922, 0.680135572180215, 0.0385616328746833, 0.10536372480232442, 0.8481227526665306, 0.39910406637895113, 0.7420436310311983, 0.5273067413490763, 0.4444018539248043, 0.06195891418585503, 0.34347072166552334, 0.7124190324509605, 0.6266784032930985, 0.5839241471066717, 0.013140421832511398, 0.851069739019065, 0.9291036934261693, 0.4457754136276317, 0.1473768396058012, 0.9625432547007423, 0.46448175537608644, 0.5448626058850728, 0.5058171104201196, 0.2681748309490154, 0.2143837601013406, 0.1478560232314542, 0.01618245105081352, 0.9784548110228287, 0.8900088606821415, 0.37892148475112464, 0.5333942121515954, 0.7565236318654025, 0.8926807618200223, 0.28556390911599805, 0.47262051257892534, 0.33295491399846733, 0.973239572240617, 0.2374906480486927, 0.5053217543779508, 0.9427258302190927, 0.39302849229100556, 0.6298699349304056, 0.22399637532606553, 0.7070801051832177, 0.8953278143402784, 0.774575617059402, 0.29239591736923715, 0.29818322476846315, 0.5654586835603503, 0.6640945988074386, 0.4379221566198031, 0.15768630903106476, 0.2337958851055223, 0.4074409685132285, 0.8603578018305627, 0.033575366266805386, 0.3011808461980805, 0.9464820511389905, 0.52383349301472, 0.06951709839958563, 0.37387280183558946, 0.6801804507524616, 0.7928357966210859, 0.19123381059208597, 0.06665462118883367, 0.26421832072745, 0.7173164662520012, 0.9378482973844969, 0.4163341412396795, 0.3279118152930922, 0.2138796310005149, 0.674958225653953, 0.02289856598847479, 0.8561985682957799, 0.12933734717282344, 0.7727939336433979, 0.347642744587568, 0.8316082832550669, 0.8404166679086241, 0.8829375402456789, 0.5312389091268362, 0.5323456947376699, 0.13409145601749953, 0.6751012861147063, 0.42731572986921096, 0.895471911828719, 0.196422105280879, 0.2663234557333977, 0.09832051121551567, 0.47283199917191265, 0.8874100823362405, 0.7012538251938549, 0.9730400331192837, 0.8838366358000025, 0.6423378906409898, 0.7729280031160116, 0.6009483708073144, 0.13926815853420094, 0.6799404843151292, 0.7597198843768425, 0.612096721591473, 0.5095997878860681, 0.843635001146996, 0.9734642775605732, 0.014112960553780645, 0.19652802739130706, 0.04655636569790373, 0.4728382846679717, 0.9930504145999673, 0.19831818165179257, 0.13367902167778417, 0.7433173385184805, 0.9345084801011293, 0.2840250596795348, 0.6091780339410426, 0.4552164471034037, 0.8228264669062693, 0.24442929366809749, 0.12313867971447234, 0.5917899611367797, 0.21387682585691886, 0.6278121772351731, 0.6392627915550315, 0.08973766541561934, 0.2209426403143176, 0.38295576273601306, 0.337504304171309, 0.4348402071906441, 0.3593622531552623, 0.8013887804939359, 0.9412337615812355, 0.31583089582427915, 0.169866118659203, 0.939856305224754, 0.16492191244145946, 0.842582403608869, 0.28245745426158303, 0.26243377442585014, 0.7244467752633834, 0.7769518516850433, 0.22977127052367258, 0.7657436913651152, 0.85422077349118, 0.8885400662610959, 0.6928936502397497, 0.463579579472346, 0.38199219171981896, 0.1427662349970854, 0.4721115960144026, 0.7795942140648114, 0.6399557872861418, 0.7369169181850352, 0.36264393588651156, 0.9566304445996091, 0.08788238563010581, 0.039255285188208935, 0.7635781582275304, 0.45810533010312604, 0.3762830432393975, 0.18910772455348993, 0.33352657050524215, 0.5810704816044636, 0.05158432622048274, 0.9777707876534065, 0.39362809080333827, 0.707322131706086, 0.9630675841882208, 0.2768874514274706, 0.6473961414989997, 0.7869501736885636, 0.27156918368841015, 0.2632702511098563, 0.7831104033547968, 0.7365491840692932, 0.18213665261032835, 0.17072042178861815, 0.29812900130549863, 0.6541249415157945, 0.8778920559575278, 0.7317844781706969, 0.10172461490227125, 0.6856026624728007, 0.9239481803606955, 0.7970673222081118, 0.31048435173485633, 0.31049960772995816, 0.5669071174072601, 0.007922263819688124, 0.14948801749827714, 0.4451100935438229, 0.9653421910318277, 0.506204671927823, 0.7819210909222817, 0.7477751307877596, 0.8566231498758416, 0.2652799632704258, 0.5603426860460745, 0.6795243763735166, 0.7661937096930079, 0.4176788103849063, 0.9277661391197546, 0.9655001857157332, 0.16162132432759801, 0.3695979739397755, 0.8331480058064442, 0.7185335889079298, 0.3940287755774468, 0.4416311301484849, 0.4944044055856785, 0.4548446784982666, 0.5745115203664226, 0.8151227984647839, 0.7688737976219848, 0.46191663269974137, 0.4328457845527892, 0.8391009787279652, 0.77014948091011, 0.9023256562195372, 0.38730408176188547, 0.4197021720091357, 0.9344049575433158, 0.544121430508849, 0.04888256222423285, 0.5692233026815687, 0.936048169124894, 0.16157848209216188, 0.649548522964841, 0.9620254700826599, 0.7620756792659292, 0.20594142247268066, 0.25748749834368356, 0.5923846622893515, 0.20901909713122022, 0.9839654844179588, 0.5078966126348342, 0.21836855365818764, 0.12028133315978634, 0.5683664165289917, 0.5343626027621154, 0.032264622874681195, 0.271516654766871, 0.38041666680035024, 0.6629189134868415, 0.6781789733461007, 0.1540050279134908, 0.3625041420396902, 0.6071152610737436, 0.786192866408356, 0.543505725238242, 0.7007240791342799, 0.06959800984226075, 0.733751418876346, 0.16009705474604716, 0.7511991168145086, 0.40355630144642496, 0.5707584100639254, 0.7365979443940325, 0.00165143050330292, 0.755592469012175, 0.24262668762478357, 0.826738909737551, 0.0008559590209536064, 0.38610326516726207, 0.2375776661734924, 0.9678353778868147, 0.40919614369477897, 0.3595870781501695, 0.5800224698986963, 0.437651587388316, 0.6102292354266295, 0.12275981536263592, 0.2242167998218056, 0.41175460508640604, 0.3596476872263698, 0.5986792135977556, 0.0015429374769064307, 0.9321501743663801, 0.6479805757515043, 0.6095366555310491, 0.4825695103418871, 0.545760316097066, 0.5936326433874819, 0.18383741340778648, 0.7554071446673046, 0.12788042338931951, 0.2862759042933005, 0.4391234575021656, 0.34795023889651067, 0.9996651336549153, 0.3719013381618547, 0.5457904862918846, 0.1007031077056672, 0.5171312091486208, 0.4242321608700939, 0.06992774366863433, 0.27558783873710213, 0.8048056544758406, 0.3686347754526114, 0.6446710320397611, 0.9860354922646822, 0.2985184925135777, 0.20030367570012048, 0.5038774919248546, 0.6690067810327778, 0.9969688178957294, 0.05492237352529651, 0.08033183965847447, 0.1372291399805011, 0.4101556522819007, 0.48604790190516406, 0.0070873200926405006, 0.1165887970088929, 0.5079113284628426, 0.4656974749945558, 0.9774622334993734, 0.2077584239690371, 0.7958316476064882, 0.54250132224639, 0.8197229950780621, 0.08437827699090274, 0.14570138610233618, 0.8031962219640595, 0.31890254994803224, 0.7951569765783646, 0.20330535257389087, 0.9530607093838326, 0.09134261407486285, 0.19531475621988753, 0.655107787649663, 0.3965870278871558, 0.43817769942720314, 0.45259427300309496, 0.7519463630169381, 0.9625232256774433, 0.1278539607896721, 0.8415189920186619, 0.4096988576509519, 0.8087005395482763, 0.8299681878788249, 0.27533367940938736, 0.5331498335735638, 0.6492528708881014, 0.9930010163192642, 0.36808127787340494, 0.3420372183164755, 0.6195282450036743, 0.41121377675384924, 0.2699459019442768, 0.9807739774607, 0.86823918198619, 0.49593164189529215, 0.12310533417533401, 0.03135148483857116, 0.9244056818654787, 0.4862951131008077, 0.16196588527502767, 0.16063381738990257, 0.7725688720925566, 0.5650332595990194, 0.5139940807195352, 0.6985146532293943, 0.9357768264300081, 0.6011218091478208, 0.05424634742282627, 0.7183611354410467, 0.49560335767250663, 0.6056324018191697, 0.8637773747852898, 0.5063380163658122, 0.02304106020510246, 0.25109886715705454, 0.218660308615612, 0.02380690259104916, 0.12261184776323468, 0.7373253566852424, 0.22726980886760625, 0.7236776378581662, 0.8500594821991676, 0.9497173214097122, 0.8990209330334425, 0.8448214930690925, 0.9148340122377658, 0.6152436801303381, 0.4005319505932424, 0.7404936206249956, 0.4762818443012805, 0.8689571716212469, 0.5631834382951183, 0.42404742605241363, 0.965089662915603, 0.2619646225413143, 0.8394110518691181, 0.9815487642686576, 0.8900810633274173, 0.5924313439021033, 0.9935969626501189, 0.38415126054741033, 0.43023602032579294, 0.9767936156023264, 0.9702974282998114, 0.7888774349302414, 0.6630488725672704, 0.8624012381128973, 0.3776089634642047, 0.47384894288789897, 0.9791831169180494, 0.1306460416552825, 0.7680221003331347, 0.14744029899474248, 0.029105204636745714, 0.17117432978524563, 0.9269607006232071, 0.4284953742420745, 0.721754886546058, 0.5343781795978445, 0.2940645009717273, 0.34206783182083994, 0.1340494128568328, 0.9684818847889461, 0.2750376478186984]

But what about different initial values? We used Python’s randint function which gives us a random value between 0 and 100000 as our initial value every time we run our program and we ran the Kolmogorov test 10000 times. Kolmogorov-Smirnov test failed for the following initial values:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Initial Value | Test Statistic | Critical Value | p-value | Test Statistic<Critical Value? |
| 47140 | 0.054027181814903014 | 0.0514692312 | 0.005609654650921693 | No |
| 28810 | 0.05608960842252225 | 0.0514692312 | 0.0035550971466579606 | No |
| 49594 | 0.05516835357536021 | 0.0514692312 | 0.004367667575460811 | No |
| 84796 | 0.05540331281880062 | 0.0514692312 | 0.004145623539864526 | No |
| 78680 | 0.05998124211327227 | 0.0514692312 | 0.0014350696970880773 | No |
| 80197 | 0.05969648592858412 | 0.0514692312 | 0.0015367278798226974 | No |
| 4791 | 0.05820211281683396 | 0.0514692312 | 0.0021891012110003194 | No |
| 29779 | 0.05694280735866297 | 0.0514692312 | 0.0029291202826374416 | No |
| 79375 | 0.05759943784754695 | 0.0514692312 | 0.0025184732833239327 | No |
| 45239 | 0.052640742057301515 | 0.0514692312 | 0.007549656546596585 | No |
| 55107 | 0.05157409674933833 | 0.0514692312 | 0.009438101198278481 | No |
| 26534 | 0.06066760874384436 | 0.0514692312 | 0.001215194212408441 | No |

Our uniform random number generator successfully passed in 98.7% of the k-s tests.

Fig: Kolmogorov-Smirnov Test result for randomly generated 1000 initial values

We rerun 1000 tests using python’s built-in random number generator and found test-statistics for k-s tests. Here is the result:

Fig: K-s Test result for randomly generated 1000 initial values using Python’s uniform random number generator

Our uniform random number generator didn’t perform worse than Python’s built-in uniform random number generator where 99.1% tests were successful

Here is the comparison of and where our test passed with an initial value of 86602, our does not deviate from much.

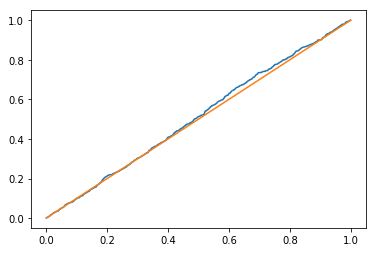


Fig: Blue line is , Orange line is . Initial value is 86602 for

And here is a graph where k-s test didn’t pass with initial value of 26534:

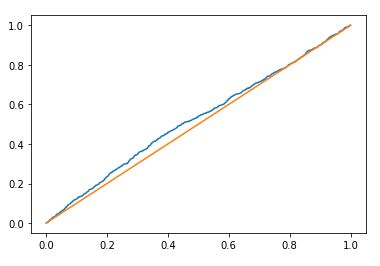


Fig: Blue line is , Orange line is . Initial value is 26534 for

Next, we will take a look at mean and variance for different distributions of our uniform random generator.

One of the most obvious things to do to test a random number generator is to average large number of values to see whether the average is close to the theoretical value. We can also find the variance for each of algorithm output for different initial values. Theoretical uniform distribution mean and variance are:

and

So, our uniform random generator must have mean close to 0.5 and variance close to 0.0833 (Taking b=1 and a=0). We ran our simulation 10000 times with each time with a different initial value between 1 and 100000. We plot the histogram of the mean and variance for each of the distribution below:

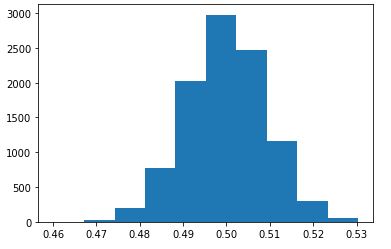


Fig: Histogram of expected values for 10000 distributions with different initial values with sample size 1000 for the uniform random generator

From the distribution we can see that most of the generated distributions has mean close to 0.5.

Next, we plot the distributions of variances of the different samples:

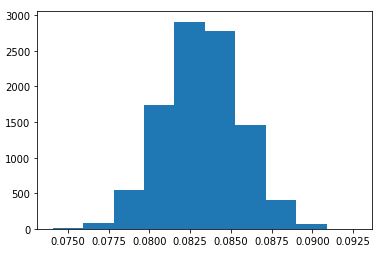


Fig : Histogram of variances for 10000 distributions with different initial values with sample size 1000 for the uniform random generator

We also plotted the histogram of means and variances of python’s built in uniform random number generator:

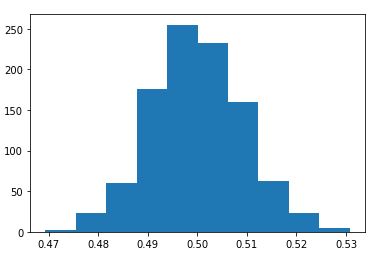


Fig : Histogram of variances for 10000 distributions with different initial values with sample size 1000 of Python’s built in uniform random number generator

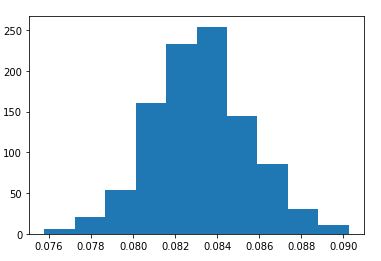


Fig : Histogram of variances for 10000 distributions with different initial values with sample size 1000 of Python’s built-in uniform random number generator

Comparing the distribution of mean and variances for our uniform random generator with Python’s uniform random generator, we can notice that they both have a normal distribution and closely follows each other. We are more assured that our uniform random generator is generating uniform random numbers. Next, we compare 95% confidence interval for both of uniform random generator:

Here are the results of mean and variance distribution:

|  |  |  |
| --- | --- | --- |
|  | Proposed Uniform Random Number Generator | Python’s built in uniform Random Number Generator |
| 95% Confidence Interval of Mean Distribution | 0.4992999991024346, 0.5004651983141434 | 0.49940226266215637, 0.5005478602619287 |
| 95% Confidence Interval of Variance Distribution | 0.08320372396850335,  0.08349011495299312 | 0.08308579078958747,  0.08337957988487102 |

From 1000 runs, we can see that the proposed uniform random number generator performed almost as well as the Python’s uniform random number generator.

So, our mean and variance test supports the claim that our uniform random number generator can generate uniform random number.

# Discussion:

Our uniform random generator performed well in Kalmogorov-Smirnov test, passing 987 out of 1000 tests with different initial values. We also generated mean and variance 1000 times using different initial value and we observed the distributions of mean and variance follow a normal distribution. In both k-s tests and mean and variance tests our uniform random generator performed almost equally as the built-in uniform random number generator which supports the claim that the random number generator can in fact generate random numbers within uniform interval.

# Conclusion:

Random number generators are complex because they are deterministic programs that give us the illusions of being nondeterministic. We cannot say how they should behave, but by conducting monte-carlo techniques like simulating our results for large number of times, we can say how they should usually behave. Without monte-carlo sampling, we couldn’t have known how our uniform random generator should behave in the long run. So, this project enabled us to learn about the importance about monte-carlo techniques as well as the properties of a good random number generator.

# Appendix:

Coding:

# -\*- coding: utf-8 -\*-

"""

Created on Sat Jan 18 12:10:02 2020

@author: Salehin

"""

import random

import math

import matplotlib.pyplot as plt

import numpy as np

import scipy.stats

n=1000

ecdf=[]

pas=0

fail=0

j=0

Initials=[]

Statistics=[]

failindex=[]

means=[]

variance=[]

means2=[]

variance2=[]

#function to find confidence interval

def mean\_confidence\_interval(data, confidence=0.95):

a = 1.0 \* np.array(data)

n = len(a)

m, se = np.mean(a), scipy.stats.sem(a)

h = se \* scipy.stats.t.ppf((1 + confidence) / 2., n-1)

return m, m-h, m+h

while(j!=1000):

l=[]

l1=[]

x0=random.randint(1,100000)

Initials.append(x0)

#uniform random number generation

for i in range(n):

x1=(7\*\*5\*x0)%(2\*\*31-1)

l.append(x1/(2\*\*31-1))

l1.append(random.uniform(0,1))

x0=x1

# plt.hist(l,edgecolor='black',linewidth=0.2)

for i in range(n):

ecdf.append((i+1)/n)

l=np.array(l)

l1=np.array(l1)

l1.sort()

l.sort()

means.append(np.mean(l,axis=0))

variance.append(np.var(l,axis=0))

means2.append(np.mean(l1,axis=0))

variance2.append(np.var(l1,axis=0))

# plt.plot(l,ecdf)

# plt.plot(ecdf,ecdf)

max3=0

for i in range(n):

x=i/n

y=(i+1)/n

g=abs(x-l[i])

h=abs(y-l[i])

maxx=max(g,h)

if(maxx>max3):

max3=maxx

Statistics.append(max3)

if(max3<1.6276/math.sqrt(n)):

pas+=1

else:

fail+=1

failindex.append(j)

j+=1