DECUS
PROGRAM LIBRARY

DECUS NO.	5-25
TITLE	A Pseudo Random Number Generator for the PDP-5 Computer
AUTHOR	P. T. Brady
COMPANY	New York University Department of Electrical Engineering
DATE	July 27, 1965
SOURCE LANGUAGE	

A PSEUDO RANDOM NUMBER GENERATOR FOR THE PDP-5

DECUS Program Library Write-up

DECUS No. 5-25

SUMMARY

The Random Number Generator Subroutine, when called repeatedly, will return a sequence of 12-bit numbers which, though deterministic, appear to be drawn from a random sequence uniform over the interval 0000 to 7777_8. Successive numbers will be found to be statistically uncorrelated. The sequence will not repeat itself until it has been called over 4 billion times.

USAGE

Calling sequence to initialize the subroutine:

JMS IR (Initialize random numbers).
Return (AC cleared upon return).

In the initializing, C(AC) are lost, but the link is preserved. IR should be called at the start of a run in order to set up the proper sequence of random numbers.

Calling sequence to obtain a random number:

JMS RN (AC and link are ignored).
Return with random number in AC. Link arbitrary.

Location: 3000-3077, binary relocatable.
Time: Approximately 0.4 msec.

METHOD

The algorithm for generating the random numbers was suggested by B. A. Tague of Bell Telephone Laboratories (private communication). The Random Number Generator actually produces a 36 bit number, denoted $\underline{x}$, and the sequence of $\underline{x}_i$ is determined as follows:

$$x_o = 1$$

$$x_n = (2^{17} + 3)\ x_{n-1} \text{ modulo 36 bits}$$

The subroutine returns only the 12 most significant bits of the 36 bit internally stored number.

When IR is called, 0000 0000 0001_8 is stored as x_o in the Random Number Generator. The first time RN is called, $(2^{17} + 3)$ is generated. Since this does not appear in the left 12 bits, 0000_8 is returned. The second number returned is 2000_8. Appendix A is a print-out of the first 500 random numbers generated by RN.

The algorithm used at Bell Telephone Laboratories differs from the one used in the PDP-5 program in two significant aspects: (1) modulo-35 bits multiplication instead of 36 bits is used, and (2) the full 35-bit product is returned as the random number, instead of just the most significant 12 bits.

STATISTICAL PROPERTIES

Although the properties of the Bell Labs random-number generator have been studied, the PDP-5 generator is sufficiently different to require a separate analysis. The generator used here is of a class known as multiplicative generators. Hull and Dobell describe a few tests which can be applied to such generators.* The tests used in this report are somewhat simpler than Hull and Dobell's, but they test the same properties.

It should be emphasized that the sequence of "ramdon numbers" generated by the subroutine is in fact completely deterministic and hence one might expect that there would exist some test for randomness which the sequence would fail. The suitability of the sequence is determined by its application. The user may wish to apply tests other than those described here, since these tests may not investigate the required properties.

All of the tests described below test the following hypotheses:

H_o(null): The numbers returned by the Random Number Generator are drawn from a sequence of independent trials from a distribution that is uniform over the interval 0000 to 7777_8.

H_i: The numbers are distributed otherwise.

SIGN (bit zero) TEST

The generator was called 2,048,000 times. The expected number of positive numbers (sign bit zero) is 1,024,000 with a standard deviation of

$$\sqrt{n\ p\ q} = \sqrt{(2{,}048{,}000)(0.5)(0.5)} = 715.5$$

We reject H_o if the measured number of positive numbers falls outside of 1,024,000 ± 1.96 s.d. for a 0.05 level test. There were actually 1,023,360 positive numbers returned, a deviation of -640 from the expected mean. (Measured success probability = 0.4997). This is well within ± 1.96 s.d., so we accept H_o.

FREQUENCY TEST OF SELECTED NUMBERS

Under H_o, with 2,048,000 trials, each of the 4096 possible 12-bit numbers should appear an average of 500 times with s.d. = 22.36. A few numbers were chosen for investigating their frequency of occurrence. If we neglect the influence of the rate of occurrence of one upon the rate of the other, we would expect 95 percent of the numbers to occur between 455 and 545 times. The results are shown in Table I.

---

*Hull, T. E. and A. R. Dobell, "Mixed Congruential Random Number Generators for Binary Machines," Jour. Ass'n for Computing Machinery, Vol. 11, Jan. 1964, p. 31.

TABLE I

FREQUENCY OF OCCURRENCE OF SELECTED NUMBERS

Number (Octal)	Occurrence (Decimal)	Within Limits?
0	501	All measured frequencies fall within 95 percent limits of H_o.
1	502	
2	500	
3	471	
4	462	
5	500	
6	522	
7	496	
10	505	
11	471	
12	513	
3777	501	
4000	504	
4001	520	
7776	503	
7777	499	

MEAN AND VARIANCE

Mean: Under H_o, the average value of the numbers returned by the generator should be 2047.5, with a s.d. of $1169.1/\sqrt{n}$. A sequence of $n = 32,768$ numbers (so that s.d. = 6.46) yielded a mean of 2047.4, almost exactly equal to the H_o population mean, and certainly within ± 1.96 s.d.'s.

Variance: The sample variance for $n = 32,768$ was measured as

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n} = 1,393,171$$

The H_o population variance is 1,398,102.

To test the variance, under H_o the statistic

$$\frac{n\, s_\mu^2}{\sigma^2} = \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{\sigma^2}$$

is distributed χ^2 with n degrees of freedom, where μ equals the H_o population mean (2047.5) and σ^2 equals the H_o population variance. For large n (32,768), the expression $\sqrt{2\chi^2} - \sqrt{2n-1}$ is distributed N(0,1). (See Hoel* p.401). Constructing a critical region for $\sigma = \sigma_o$ vs. $\sigma \neq \sigma_o$, we reject H_o if ns_μ^2 falls outside the limits 32,268 to 33,272. The measured value for $\frac{ns_\mu^2}{\sigma^2}$ is

$$\frac{ns_\mu^2}{\sigma^2} = 32,652$$

Again, we accept the null hypothesis.

AUTOCORRELATION

A common requirement of random number sequences is that the numbers be sequentially independent. A 0.05 level test for <u>correlation</u> was constructed. If the numbers are sequentially <u>independent</u>, (i.e., if H_o is true), this correlation test will fail at most five percent of the time. Since, however, a set of numbers can be sequentially uncorrelated and yet not be independent, this test is relatively weak.

---

*Hoel, P. G., Introduction to Mathematical Statistics, third edition, John Wiley and Sons, New York, 1962.

The autocorrelation function of 32,768 random numbers was computed for shifts from 1 to 99, inclusive.* (99 numbers were generated prior to the 32,768 so that the shift product could be formed for all the 32,768 numbers). One would expect that the average product would cluster about $E(x^2)$ which, for an independent sequence, yields

$$E\,(x^2) = [E(x)]^2 = (2047.5)^2 = 4{,}192{,}256.$$

This in fact did occur. For example, $\overline{x_i}^{2}$ for shifts of 1, 2, and 3 were 4,192,555, 4,189,914, and 4,196,667, respectively.

Hoel (p. 165) defines a <u>correlation coefficient</u>, <u>r</u>, which for an autocorrelation function becomes:**

$$r = \frac{\dfrac{x_i\ x_{i+\tau}}{n} - \overline{x_i}^{2}}{\dfrac{x_i^{2}}{n} - \overline{x_i}^{2}}.$$

where $\{x_1\}$ is one sequence and $\{x_1+\tau\}$ is the shifted sequence. If the sequences are uncorrelated, then the quantity

$$Z = \tfrac{1}{2}\ \ln\left(\frac{1 + r}{1 - r}\right)$$

will be distributed approximately normally with zero mean and variance $1/(n-3)$, where n, in this particular example, equals 32,768. Constructing a 0.05 level test, the H_o acceptance region for <u>r</u> becomes -0.010816 to +0.01088. This results in the acceptance region for $\overline{x_i\ x_{i+\tau}}$ to be from 4,176,779 to 4,207,005.

When the average products $\overline{x_i\ x_{i+\tau}}$ were examined for $1 \le \tau \le 99$, the following values of τ yielded products out of the 95 percent acceptance region:

---

* The autocorrelation functions for a shift of zero is related to the variance and is discussed in Section 4.3.

** An approximation is made that $\overline{x}_i = \overline{x}_{i+\tau}$, and that $\dfrac{x_i^2}{n} = \dfrac{x(x_i+\tau)^2}{n}$.

TABLE II

Values of τ for which the Autocorrelation Function Fails a 0.05 Level Test

τ	$\overline{x_i \; x_{i+\tau}}$
21	4,207,225
24	4,172,528
30	4,208,613
47	4,207,888
57	4,171,403
59	4,173,423
68	4,208,418
98	4,175,611

Acceptance region:

from 4,176,779
to 4,207,005

There are several ways to interpret these results. One way is to state that the sequence of numbers appears to be correlated for the 8 listed values of τ. If the user is concerned about correlation at $\tau = 30$, for example, he would be justified in rejecting this random number generator. A possible alternative would be to write another program using, say, $2^{18} + 3$ as the multiplier (instead of $2^{17} + 3$).

The author's interpretation of the data is that if 99 correlation coefficients are chosen from a population in which 5 percent exceed certain limits, then one would expect about 5 "failures" in the 99. In fact, with $p = 0.05$ and $n = 99$, 93.5 percent of the experiments will yield between 2 and 9 failures* Having obtained 8 failures, we are thus within reasonable bounds for H_o.

In summary, H_o is accepted on the basis of the measured autocorrelation function.

The autocorrelation function is, unfortunately, a gross statistic and certain subtile interdependencies may still exist. For example, it is conceivable that if the generator returns the value of zero, the probability that the next number will be zero would be much less than 1/4096. For such a specialized case, it is recommended that the user develop his own test for the sequence.

THE UNIFORM (0, 1) DISTRIBUTION

Numbers are sometimes interpreted, in the PDP-5 computer, as a signed fraction with the binary point to the right of the sign bit. The highest possible positive number is thus $3777_8 \approx 0.9995$. The following calling sequence will cause the random number generator to return positive fractions uniform over (0, 1):

```
JMS  RN
SPA
CMA
```

The one's complement CMA instruction establishes a one-to-one correspondence between all negative numbers and all positive numbers. Appendix B is a print-out of the first 500 random numbers, interpreted as fractions, in which negative numbers were transformed with the above sequence of instructions.

SEQUENCES OF LONG RUNS

The user may wish to use several different long sequences of random numbers. One sequence, for example, would use numbers 0 to 1 million, the next, 5 million to 6 million, then 10 million to 11 million, etc. It can be very time consuming to call the generator 10 million times just to initialize the desired sequence.

---

*Weintraub, S., Tables of the Cumulative Binomial Probability Distribution, Collier-MacMillan, London 1963, p. 800.

A better way is to write a short program which will call RN many millions of times, and after each million numbers, type out the contents of registers 3075, 3076, and 3077. (See the symbolic dump in Appendix C). From then on, the generator can be initialized to any starting point simply by putting the appropriate 36 bit number into these registers.

Appendix A: Tabulation of the First 500 Random Numbers Returned by RN

```
0000 2000 2000 4000 4003 6013 6047 0210 0715 5001
3722 4337 5326 2453 4571 7117 6236 7346 1726 2352
3365 0605 3177 6520 0542 2174 5164 5133 6022 7462
5210 5154 5703 6301 7252 4457 7433 5772 5751 7654
7325 7762 3052 4565 6505 7175 5605 3262 6204 1316
6063 1762 2733 1637 6403 0572 7676 2446 6470 2366
1316 4057 1723 0516 0546 0243 1463 0567 4000 3322
6350 2007 0003 7721 5314 3153 0523 1061 6464 7577
1046 2555 4464 5142 3567 1533 6353 1521 5641 5747
5301 4540 1571 2166 3600 4323 6560 2465 1116 1370

4420 7650 4136 4511 4147 1324 0532 1243 7652 5075
6155 1152 5243 7033 0343 5137 1075 2024 0113 0417
7666 4275 1410 3604 6723 1511 1514 5065 5624 6631
2541 4740 3325 2037 3467 5064 4103 4701 6044 2422
6441 2451 2307 1067 7111 2702 6777 2042 3321 3665
6344 5773 7726 3466 4265 3535 4712 6165 7246 5703
7272 1202 3201 2156 6014 2152 5016 4227 2412 2354
6054 0270 1301 7015 3611 1274 4225 7335 3764 6757
4006 3273 6053 0155 0414 3156 4443 0365 7577 4133
1255 4330 4367 5061 4162 5360 0640 5515 0455 2524

0535 6074 4442 6250 2700 2221 1652 0335 0065 6564
5336 6036 1340 4061 1502 6720 0621 0424 2131 2147
7502 3753 1061 4742 1622 0167 7251 3706 3256 3035
2220 4132 2416 3447 4547 6006 7606 2354 6720 5217
7407 7442 7614 4425 3224 4662 5373 7233 0317 2544
2421 0735 4622 6243 0657 6135 3377 7266 2514 7535
4200 6273 5741 4254 2440 2262 5014 3004 2655 7746
4123 3326 5031 1417 1571 6514 6204 3553 3732 0127
3540 6456 4263 0623 3440 0224 7530 5325 1345 1740
4457 4064 5224 4635 3172 2130 5703 1562 3321 6744

0772 1327 5471 6707 4651 0766 0323 3510 0106 2026
1021 5635 7416 2716 3531 5722 0314 7136 5411 2350
3032 6166 7727 1333 7224 0705 1550 4201 0132 4616
1051 5372 5147 7645 6071 0206 0445 3040 3166 1641
3634 0400 6604 6024 5322 4063 5713 0366 3644 4457
7122 3501 3244 5213 7570 0757 6135 6310 4545 4526
7566 6272 6463 0247 4434 2311 3272 0511 3452 1547
7757 2374 5173 3004 0706 4167 7327 0330 5601 0551
6352 2310 2161 1636 4671 6677 3371 1037 4773 6310
7333 5031 1332 5075 7700 3136 2157 2512 1723 5524

6607 7464 6574 6021 5407 4616 1030 5215 4166 3716
7246 2646 5015 6570 1126 4314 5675 5504 2755 1052
1651 3172 0747 2431 4567 7354 0122 7643 3352 0301
1113 5373 4471 4201 6011 7646 6623 4231 2556 3457
3072 4260 4021 3067 4255 0037 7237 5246 4110 2733
2231 5337 1724 4441 0121 1667 7560 7454 6031 2007
3503 5524 7231 2643 5156 7730 6462 2222 0243 7261
7154 3106 7734 5254 6514 1667 5232 1054 4633 5031
1037 2730 5763 3434 5427 3221 7626 3145 3026 7364
7366 3455 0143 2673 5343 0701 2204 7522 7506 2673
```

Appendix B: Tabulation of the First 500 Random Numbers interpreted as Fractions

.0000	.5000	.5000	.9995	.9980	.4941	.4804	.0664	.2250	.7490
.9775	.8906	.6450	.6459	.8154	.2109	.4223	.1372	.4794	.6142
.8696	.1899	.8120	.3354	.1728	.5605	.6928	.7050	.4907	.1000
.6831	.6967	.5292	.4052	.1665	.8515	.1113	.5024	.5107	.0405
.1455	.0063	.7705	.8173	.3408	.1884	.5595	.8369	.4350	.3505
.4746	.4931	.7319	.4526	.3730	.1845	.0317	.6435	.3471	.6201
.3505	.9765	.4780	.1630	.1748	.0795	.3999	.1831	.9995	.8525
.3862	.5034	.0014	.0224	.6499	.8022	.1655	.2739	.3491	.0625
.2685	.6782	.8491	.7016	.9331	.4194	.3847	.4145	.5458	.5117
.6552	.8276	.4340	.5576	.9375	.8964	.3198	.6508	.2880	.3710

.8666	.0424	.9536	.8388	.9492	.3535	.1689	.3295	.0415	.7197
.4462	.3017	.6699	.2363	.1108	.7031	.2797	.5097	.0366	.1323
.0356	.9072	.3789	.9394	.2714	.4106	.4121	.7236	.5522	.2998
.6723	.7651	.8540	.5151	.9018	.7241	.9567	.7802	.4819	.6337
.3583	.6450	.5971	.2768	.2138	.7197	.2500	.5166	.8520	.9633
.3881	.5019	.0200	.9013	.9111	.9204	.7758	.4423	.1684	.5292
.1586	.3134	.8129	.5537	.4936	.5517	.7426	.9257	.6298	.6152
.4780	.0898	.3442	.2431	.9418	.3417	.9267	.1416	.9941	.2578
.9965	.8413	.4785	.0532	.1308	.8037	.8574	.1196	.0625	.9550
.3344	.8940	.8789	.7255	.9438	.6323	.2031	.5869	.1469	.6660

.1704	.4702	.8579	.4174	.7187	.5708	.4580	.1079	.0258	.3178
.6411	.4848	.3593	.9755	.4072	.2729	.1958	.1347	.5434	.5502
.0922	.9897	.2739	.7641	.4462	.0581	.1669	.9716	.8349	.7641
.5703	.9555	.6318	.8940	.8242	.4965	.0590	.6152	.2729	.6796
.1210	.1079	.0561	.8642	.8222	.7875	.6269	.1738	.1010	.6738
.6333	.2329	.8032	.4199	.2104	.4541	.8745	.1606	.6621	.0791
.9370	.4082	.5146	.9155	.6406	.5869	.7436	.7519	.7094	.0122
.9589	.8544	.7373	.3823	.4340	.3374	.4350	.9272	.9814	.0424
.9218	.3520	.9121	.1967	.8906	.0722	.0815	.6455	.3618	.4843
.8515	.9741	.6772	.7978	.8095	.5429	.5292	.4306	.8520	.2631

.2470	.3549	.5966	.2773	.7919	.2451	.1030	.9101	.0341	.5107
.2583	.5478	.1176	.7255	.9184	.5219	.0996	.2036	.6201	.6132
.7626	.4418	.0195	.3569	.1772	.2211	.4257	.9365	.0439	.8051
.2700	.6274	.6992	.0439	.4716	.0654	.1430	.7656	.8076	.4536
.9511	.1250	.3100	.4897	.6469	.9746	.5253	.1201	.9550	.8515
.2094	.9067	.8300	.6816	.0659	.2416	.4541	.4018	.8251	.8325
.0668	.4086	.3496	.0815	.8608	.5981	.8408	.1606	.8955	.4252
.0078	.6230	.6894	.7519	.2216	.9414	.1445	.1054	.5615	.1762
.3852	.5976	.5551	.4521	.7841	.2812	.8715	.2651	.7519	.4018
.1425	.7373	.3564	.7197	.0307	.7958	.5541	.6611	.4780	.5834

.3085	.0991	.3139	.4912	.6210	.8051	.2617	.6806	.9418	.9755
.1684	.7060	.7431	.3159	.2919	.8999	.5322	.5913	.7407	.2705
.4575	.8095	.2377	.6372	.8164	.1342	.0400	.0449	.8642	.0942
.2866	.6269	.8466	.9365	.4951	.0434	.3027	.9248	.6787	.8979
.7783	.9135	.9912	.7768	.9150	.0151	.1718	.6684	.9643	.7319
.5747	.6406	.4785	.8583	.0395	.4643	.0698	.1030	.4873	.5034
.9077	.5834	.1748	.7045	.6958	.0190	.3500	.5712	.0795	.1630
.1967	.7841	.0170	.6655	.3374	.4643	.6743	.2714	.7988	.7373
.2651	.7304	.5058	.8886	.6132	.8208	.0512	.7993	.7607	.1303
.1293	.8969	.0483	.7163	.6386	.2192	.5644	.0844	.0903	.7163

APPENDIX C

SYMBOLIC DUMP OF THE RANDOM NUMBER GENERATOR

```
      3000,3077S
IR -> 3000 7402 HLT
      3001 5263 JMP 3063
RN -> 3002 7402 HLT
      3003 7200 CLA
      3004 1277 TAD 3077
      3005 7004 RAL
      3006 0271 AND 3071
      3007 3274 DCA 3074
      3010 1276 TAD 3076
      3011 0270 AND 3070
      3012 1274 TAD 3074
      3013 7006 RAL,RAT
      3014 7006 RAL,RAT
      3015 7004 RAL
      3016 3272 DCA 3072
      3017 1277 TAD 3077
      3020 0270 AND 3070
      3021 7006 RAL,RAT
      3022 7006 RAL,RAT
      3023 7004 RAL
      3024 1276 TAD 3076
      3025 3273 DCA 3073
      3026 7430 SNL,SKP
      3027 2272 ISZ 3072
      3030 7000 NOP
      3031 1277 TAD 3077
      3032 7104 CLL,RAL
      3033 7430 SNL,SKP
      3034 2273 ISZ 3073
      3035 7410 SKP
      3036 2272 ISZ 3072
      3037 7000 NOP
      3040 7100 CLL
      3041 1277 TAD 3077
      3042 3277 DCA 3077
      3043 1276 TAD 3076
      3044 7004 RAL
      3045 7430 SNL,SKP
      3046 2272 ISZ 3072
      3047 7000 NOP
      3050 7100 CLL
      3051 1273 TAD 3073
      3052 3276 DCA 3076
      3053 1275 TAD 3075
      3054 7004 RAL
      3055 1275 TAD 3075
      3056 1272 TAD 3072
      3057 3275 DCA 3075
      3060 1275 TAD 3075
      3061 5602 JMP* 3002
      3062 0000 not used
```

Multiplies the number in 3075, 3076, and 3077 (a 36-bit number) by $(2^{17} + 3)$ and replaces the previous number by the new product, modulo 36 bits.

Most Significant 12 bits of $(2^{17})\chi_{n-1}$ goes into 3072

(Link is zero)

Middle 12 bits of $(2^{17}+1)\chi_{n-1}$ in 2073

overflow into 3072

Add $2\chi_{n-1}$ to Middle 12 bits

Overflow into 3072

3077 now contains least 12 bits of $3\chi_{n-1}$, or $(2^{17} + 3)\ \chi_{n-1}$ [The 2^{17} does not appear in least 12 bits]

Final 12 middle bits

Final most significant 12 bits

```
       3063 7201 CLA,IAC
       3064 3277 DCA 3077
IR     3065 3276 DCA 3076
       3066 3275 DCA 3075
       3067 5600 JMP* 3000
       3070 0177 \
       3071 7400 /  Constants
       3072 0000 \
       3073 0000  > Temp. Storage
       3074 0000 /
       3075 0000 most    \       36-bit stored
       3076 0000 middle   >      random number
       3077 0000 least   /
```