TM No. 921244

NAVAL UNDERSEA WARFARE CENTER
DETACHMENT NEW LONDON
NEW LONDON, CONNECTICUT

Technical Memorandum

WHITENESS IN RANDOM NUMBER GENERATORS

Date: 28 December 1992

Prepared by: [illegible]
Alfredo J. Edmonds
System Development Division
Submarine Sonar Department

[illegible]
James A. Ionata
System Development Division
Submarine Sonar Department

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **28 DEC 1992**	2. REPORT TYPE **Technical Memorandum**	3. DATES COVERED **28-12-1992 to 28-12-1992**
4. TITLE AND SUBTITLE **Whiteness in Random Number Generators**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Alfredo Edmonds; James Ionata**		5d. PROJECT NUMBER **A17653**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Undersea Warfare Center Division,Newport,RI,02841**		8. PERFORMING ORGANIZATION REPORT NUMBER **TM 921244**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **NAVSEA 06U2**		10. SPONSOR/MONITOR'S ACRONYM(S) **NAVSEA 06U2**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
NUWC2015

14. ABSTRACT
Three different pseudo-random number generators were examined for use in a sonar detection simulation using a whiteness test. The fluctuation of the whiteness measure and the probability that an acceptable sequence of numbers could be produced were studied. The ran1 generator, initially thought to be suitable, was found to be unacceptable for our use. Random was the recommended generator.

15. SUBJECT TERMS
pseudo-random numbers; whiteness test

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**22**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

ABSTRACT

Three different pseudo-random number generators were examined for use in a sonar detection simulation using a whiteness test. The fluctuation of the whiteness measure and the probability that an acceptable sequence of numbers could be produced were studied. The *ran1* generator, initially thought to be suitable, was found to be unacceptable for our use. *Random* was the recommended generator.

ADMINISTRATIVE INFORMATION

This memorandum was prepared under Project No. A17653, "AN/BQR-22A EC 15," principal investigator John Sanchis (Code 2152). The sponsoring activity is the NAVSEA 06U2, project manager Thomas Ronaldi.

The authors of this memorandum are located at the Naval Undersea Warfare Center, Detachment New London, CT 06320.

ACKNOWLEDGMENT

The authors acknowledge that the success of this research represents the cumulative efforts and cooperation among many individuals and organizations; specifically the participation of Dr. Albert Nuttall at NUWC/NL (Code 302).

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

INTRODUCTION

A stochastic process[1] is a set of random variables having a joint probability distribution.[2] The process has particular properties by which it can be identified, such as the case of the Markov[3] and Poisson[4] processes. If the random variables are generated by a deterministic algorithm, they are called pseudorandom variables[5] because their distributions are given by a probability function that can be implemented in a computer.

Several algorithms have been developed to simulate random number generators. Most fall short of expectations because their randomness is dependent upon the finite number of states of the machine and/or their deterministic approach. In general, there will be a point when the output of the algorithm will become periodic. If the output becomes predictable too soon, the algorithm is not useful for applications involving several million numbers. In fact, the algorithm can then be known as a "bad" random number generator.

Different methods have been used to determine the efficiency and effectiveness of the algorithm and its randomness. Among the methods used are the uniformity, serial correlation and spectral tests. There are many other tests that can be performed on the algorithm but, in general, the effectiveness and efficiency of the algorithm will depend mostly on its particular application.[6]

This study was conducted as part of a larger project to simulate the performance of a sonar detection algorithm. The detection algorithm was designed to work when the background noise has a Gaussian distribution and is stationary. A uniform random number generator was required as the first step in simulating such noise. The uniformly distributed deviates were then transformed using the polar method.[7] The performance of the uniform random number generator used was important because of the large number (on the order of 100 million) of independent deviates required for the simulation.

THEORETICAL ANALYSIS

To satisfy the requirements of the simulation, the generator had to pass a statistical periodicity test so that there would be enough random data available to simulate the task. The outputs of candidate random number generators were exposed to a whiteness test[8] to determine their effectiveness. In this context, "whiteness" refers to the signal's autocorrelation function and error measure. The whiteness performance test was implemented in both the time and frequency domains. The time domain approach, however, was quickly abandoned because it took longer than the frequency implementation to obtain an output.

Three random number generators were tested: *ran1* (App. A), *rand*[9] and *random.*[10] Uniform real random numbers in the range of $\left(-\frac{1}{2}, \frac{1}{2}\right)$ were generated by the algorithms and placed in an array. These numbers were converted to complex, frequency domain numbers, through an FFT routine. Their magnitudes (X_m) were calculated and used for the determination of the threshold test (Q) of the signal. Q_1, the measure of whiteness[11] (or simply called whiteness) could then be mathematically expressed as:

where
$$Q_1 = Q - 1$$

$$Q = M * \frac{\sum_{m=0}^{M-1} |X_m|^4}{(\sum_{m=0}^{M-1} |X_m|^2)^2},$$

and M is the number of discrete Fourier Transform points, and must be greater than or equal to twice the number of data points K. The value of K is the amount of sequential random numbers needed from the random number generator to make up one trial. For practical reasons, the value of K was chosen to be of form 2^n, so that M, after executing the FFT routine, would have its entire array filled with useful data. For example, if K = 262,144 data points and there are 1000 trials, the algorithm will have to generate 262,144,000 numbers. Had M been chosen to be greater than twice the number of data points K there would have been some points in M with unkown data.

The behavior of the algorithms' whiteness for increments of $K = 2^6$ to $K = 2^{20}$ was recorded, with the emphasis placed on the value of their expectations. Expectation refers to the mean of the set of sequential random numbers at that particular increment, after a number of trials had been produced. Whenever possible, the total number of trials was one thousand, allowing for an accurate calculation of the expectation. For this experiment's purposes, about 100 million generated points were sufficient.

According to previous work,[12] the expectation of the signal should not be significantly greater than the whiteness figure of merit, Q_1, of 1.5. Ideally, the measure of whiteness should approach zero as K increases. However, it tends to approach one after the algorithm's output has been normalized. If the algorithm's expected value surpasses the threshold, then the generator is considered inadequate for this research's specific purpose.

The maximum and minimum values of whiteness and the probability of the measure being above the threshold were also recorded to further analyze the behavior of the algorithms. Fluctuation of the signal between maximum and minimum describes the stability of the algorithm and it is defined as the difference between the highest possible whiteness of all trials and the lowest possible whiteness of all trials. The probability describes how often the algorithm is expected to rise above the figure of merit under the same conditions that it was exposed in this evaluation. The probability is defined as the number of times the algorithm's output is above the figure of merit divided by the total number of trials under which the algorithm was tested at the particular n increment.

WHITENESS ANALYSIS

The routine in appendix B, written in C language in the Unix environment of the Sun workstation, was used to obtain the results of running *ran1*, *rand* and *random* depicted in Tables 1 through 3 respectively. Only the name of the candidate random number generator was changed in the routine so that there would be no other discrepancies when the results were compared. These data are also shown in Charts 1 through 3, which describe the algorithms' whiteness as K (data points) increases. The small boxes on the charts represent actual data. Graph smoothing was carried out by polynomial interpolation. *Ran1*, *rand* and *random* were approximated with 6th, 5th and 3rd order polynomial equations respectively.

Sometimes the algorithms were running for several days before the result of at least one trial was attained. Less than one thousand trials were collected for those cases and though the average whiteness is less precise, the data may still be used to measure the performance of the algorithm because the algorithm generated more than 1 million points.

The tabulated data prompted the observation that *rand*'s fluctuation seems to be slightly approaching zero faster than *ran1*'s and *random*'s fluctuations at the beginning, but as K increases its fluctuation becomes higher than that of both algorithms. Although it may seem contradictory, note that while *rand*'s fluctuation is high, of the three algorithms, it has the lowest probability of having the measure of whiteness above the figure of merit, which only means that *rand*'s whiteness are within a wider range of values than the other two. The results also denoted *ran1*'s average whiteness at $K = 2^{20}$ points rising above the threshold, while the other two were yet to approach the limit. In fact, *ran1*'s probability of failure at that point was 100%!

SUMMARY

According to the computer simulation, the test results showed that *ran1*'s average whiteness exceeded the threshold value. The results of the three candidate random number generators may be compared using Chart 4. Even after 1,048,576 points, only *ran1* would have completely failed the test. Therefore, *ran1* cannot be recommended for this particular application. *Ran1* is unreliable when attempting to generate greater than 262,144 random numbers.

When fluctuation becomes a major consideration, *random* is the best choice of the three. The results showed that as K increases, for every one thousand trials, there was almost no significant difference in the outputs of *random*. And, if the concern is for the number of times the algorithm's output is higher than the figure of merit, then the results demonstrated *rand* would satisfy this requirement better than the other algorithms.

Even though these three random number generators were tested for more than 524,288 data points, it does not necessarily mean that they will work successfully once they are employed. This research only set the basis for future analysis concerning random number generation. While there are many other available tests, as it was mentioned before, there may also be other algorithms which can perform more effectively and efficiently than those analyzed here. However, no computer program can generate truly random numbers. "Good" random number generators are those whose outputs have been tested and found acceptable for particular applications. Future work in this area should include a whiteness test on more random number generators as well as their measured performance in different tests, such as χ^2 and spectral tests.

REFERENCES

[1] J. L. Doob, "Stochastic Processes," New York, 1953 (Wiley Classics Library Edition Published 1990.)

[2] H. Cramer, "Random Variables and Probability Distributions," University Press, Cambridge, 1970.

[3] D.P. Heyman, M.J. Sobel, Eds., "Stochastic Models, Vol. 2 of Handbooks in Operations Research and Management Science," Elsevier Science Publishers B.V., North Holland, 1990.

[4] A.B. Clarke, R.L. Disney, "Probability and Random Processes for Engineers and Scientists," John Wiley & Sons, New York, 1970.

[5] H. Niederreiter, "Random Number Generation and Quasi-Monte Carlo Methods," SIAM, 1992.

[6] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling,"Numerical Recipes: The Art of Scientific Computing," University Press, Cambridge, 1970. pp 191-192.

[7] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, "Numerical Recipes: The Art of Scientific Computing," University Press, Cambridge, 1970. pp 202-203.

[8] A. Nuttall, "On Generation of Random Numbers with Specified Distributions or Densities," NUSC TR 6843, Detach. New London, 1 December 1982.

[9] Sun Workstations, "SunOs Reference Manual Vol. II, 3. C Library Functions, Sun Release 4.1," Revision A of 27 March, 1990, USA.

[10] Sun Workstations, "SunOs Reference Manual Vol. II, 3. C Library Functions, Sun Release 4.1," Revision A of 27 March, 1990, USA.

[11] A. Nuttall, "On Generation of Random Numbers with Specified Distributions or Densities," NUSC TR 6843, Detach. New London, 1 December 1982.

[12] A. Nuttall, "Statistics of a White Measure," NUSC-NL TR 10237, 9 December 1992.

RAN1 RANDOM NUMBER GENERATOR

K data points	M FFT size	trials	whiteness	minimum whiteness	maximum whiteness	fluctuation	probability (%)
64	128	1000	.973	.361	2.391	2.029	5.000
128	256	1000	.979	.578	1.919	1.341	1.800
256	512	1000	.996	.686	1.632	.946	.400
512	1024	1000	.993	.766	1.567	.801	.200
1024	2048	1000	.999	.810	1.405	.595	.000
2048	4096	1000	1.001	.869	1.224	.355	.000
4096	8192	1000	1.002	.890	1.146	.256	.000
8192	16384	1000	1.002	.911	1.093	.182	.000
16384	32768	1000	1.003	.941	1.060	.120	.000
32768	65536	1000	1.007	.966	1.048	.082	.000
65536	131072	1000	1.020	.995	1.047	.052	.000
131072	262144	1000	1.044	1.026	1.063	.037	.000
262144	524288	1000	1.265	1.255	1.273	.018	.000
524288	1048576	1000	2.053	2.041	2.061	.020	100.000
1048576	2097152	4	3.342	*	*	*	*

* not available

Table 1

RAND -UNIX'S RANDOM NUMBER GENERATOR

K data points	M FFT size	trials	whiteness	minimum whiteness	maximum whiteness	fluctuation	probability (%)
64	128	1000	.982	.461	2.151	1.690	4.900
128	256	1000	.995	.592	1.974	1.383	1.700
256	512	1000	1.000	.646	1.859	1.213	.300
512	1024	1000	1.000	.714	1.412	.698	.000
1024	2048	1000	1.001	.774	1.286	.512	.000
2048	4096	1000	.997	.867	1.211	.345	.000
4096	8192	1000	.998	.908	1.128	.220	.000
8192	16384	1000	1.000	.929	1.094	.166	.000
16384	32768	1000	1.000	.940	1.074	.134	.000
32768	65536	1000	1.000	.957	1.037	.080	.000
65536	131072	1000	1.000	.976	1.031	.054	.000
131072	262144	1000	1.000	.981	1.020	.039	.000
262144	524288	1000	1.000	.987	1.015	.028	.000
524288	1048576	1000	1.000	.988	1.010	.022	.000
1048576	2097152	3	1.000	*	*	*	*

* not available

Table 2

RANDOM -UNIX'S RANDOM NUMBER GENERATOR

K data points	M FFT size	trials	whiteness	minimum whiteness	maximum whiteness	fluctuation	probability (%)
64	128	1000	.976	.500	3.085	2.585	4.200
128	256	1000	.998	.578	2.132	1.554	2.900
256	512	1000	.991	.690	1.569	.879	.400
512	1024	1000	.992	.723	1.430	.706	.000
1024	2048	1000	.996	.784	1.251	.468	.000
2048	4096	1000	.999	.852	1.182	.330	.000
4096	8192	1000	.998	.882	1.133	.251	.000
8192	16384	1000	.999	.915	1.080	.166	.000
16384	32768	1000	1.000	.951	1.051	.100	.000
32768	65536	1000	1.000	.958	1.044	.086	.000
65536	131072	1000	1.000	.970	1.030	.060	.000
131072	262144	1000	1.000	.981	1.020	.039	.000
262144	524288	1000	.999	1.000	1.018	.018	.000
524288	1048576	1000	1.000	.996	1.007	.012	.000
1048576	2097152	6	1.000	*	*	*	*

* not available

Table 3

RAN1 - RANDOM NUMBER GENERATOR

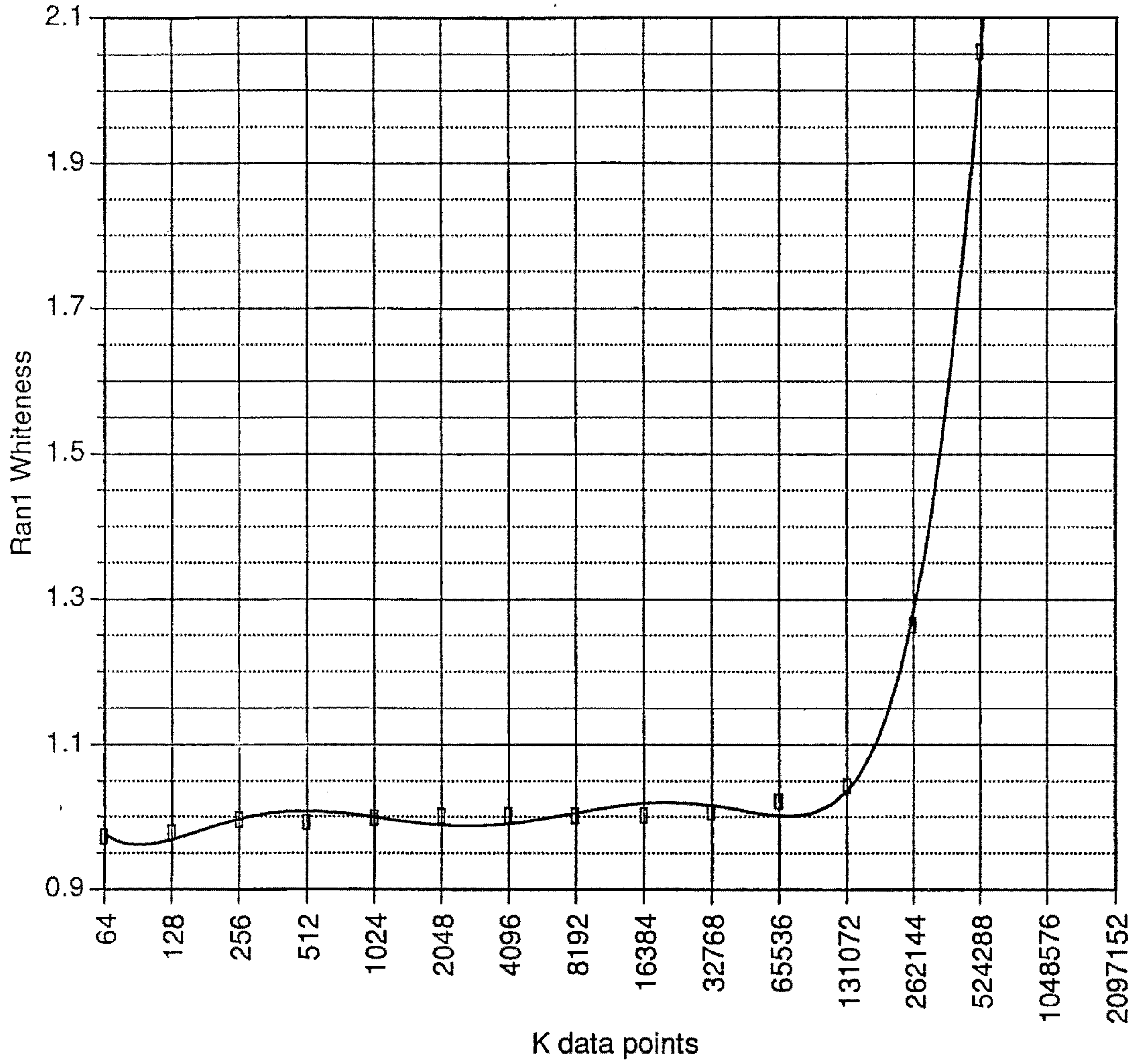

CHART 1

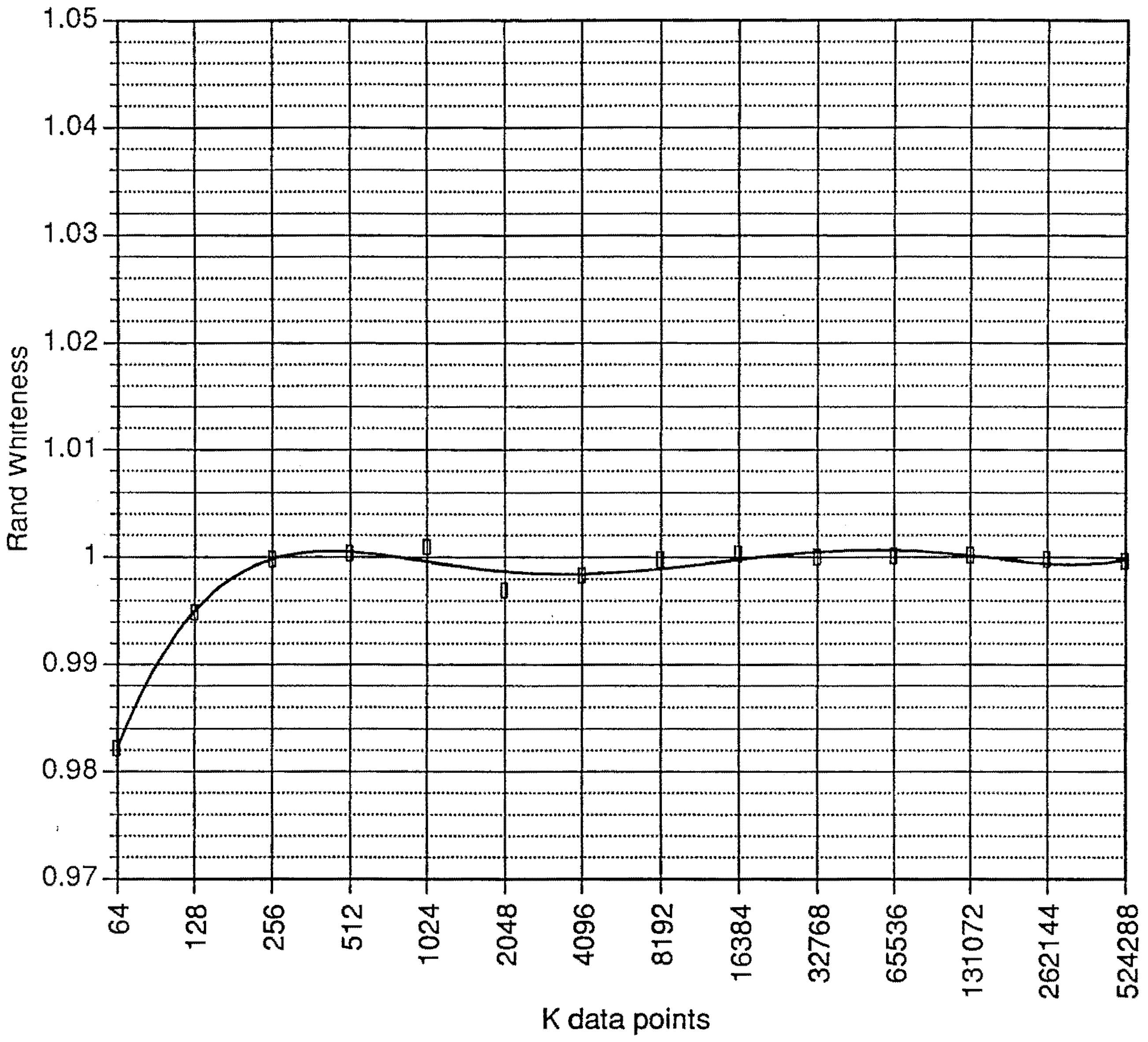
1.05
1.04
1.03
1.02
1.01
1
0.99
0.98
0.97
Rand Whiteness
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
K data points

CHART 2

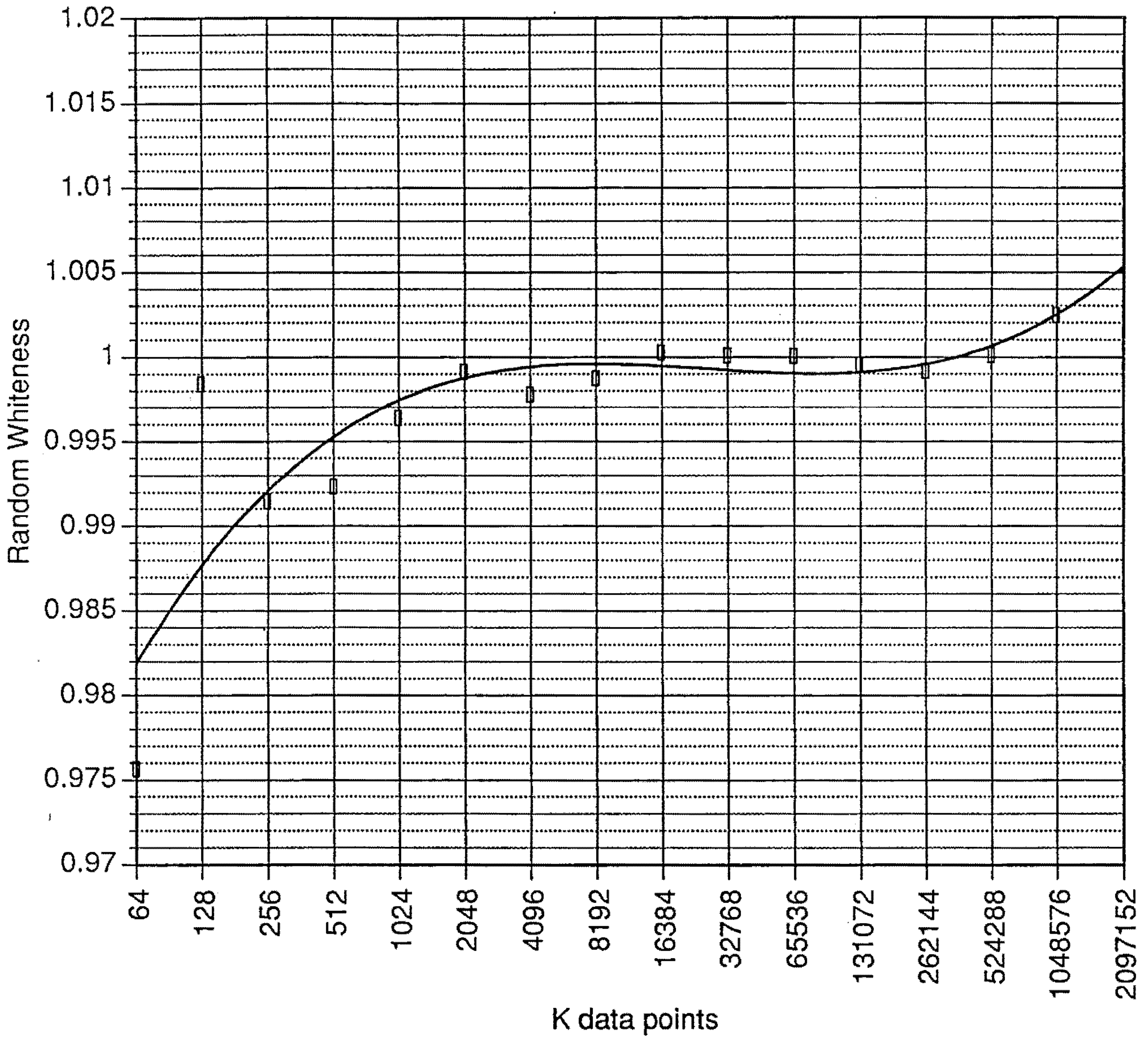
Random Whiteness
1.02
1.015
1.01
1.005
1
0.995
0.99
0.985
0.98
0.975
0.97
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
1048576
2097152
K data points

CHART 3

ALGORITHMS' WHITENESS

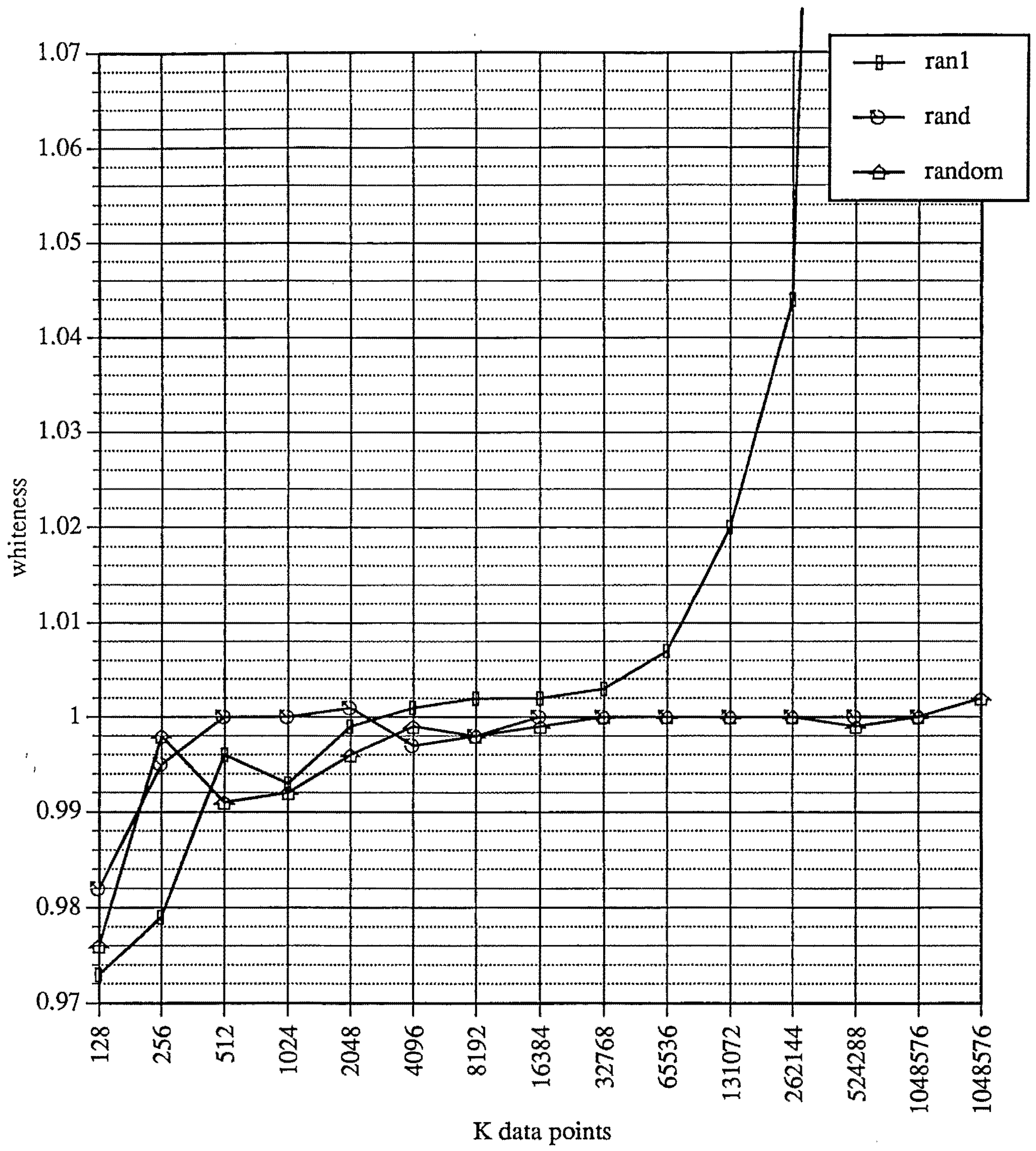

Chart 4

APPENDIX A

```
/***********************************************************************
* Ran1(idum) to be tested on whiteness
* W.H. Press, B.P. Flannary, S.A. Teukolsky, and W.T. Vetterling,
* "Numerical Recipes: The Art of Scientific Computing", 1986, p196.
***********************************************************************/

#define M1 259200
#define IA1 7141
#define IC1 54773
#define RM1 (1.0/M1)
#define M2 134456
#define IA2 8121
#define IC2 28411
#define RM2 (1.0/M2)
#define M3 243000
#define IA3 4561
#define IC3 51349
#include <stdio.h>

double ran1(idum)
int idum;
{
     static long ix1,ix2,ix3;
     static double r[98];
     double temp;
     static int iff=0;
     int j;

     if (idum < 0 || iff == 0)
          {
          iff=1;
          ix1=(IC1-idum) % M1;
          ix1=(IA1*ix1+IC1) % M1;
          ix2=ix1 % M2;
          ix1=(IA1*ix1+IC1) % M1;
          ix3=ix1 % M3;
          for (j=1;j<=97;j++) {
               ix1=(IA1*ix1+IC1) % M1;
               ix2=(IA2*ix2+IC2) % M2;
               r[j]=(ix1+ix2*RM2)*RM1;
          }

     }
     ix1=(IA1*ix1+IC1) % M1;
     ix2=(IA2*ix2+IC2) % M2;
     ix3=(IA3*ix3+IC3) % M3;
     j=1 + ((97*ix3)/M3);
     if (j > 97 || j < 1) printf("RAN1: This cannot happen.\n");
     temp=r[j];
     r[j]=(ix1+ix2*RM2)*RM1;
     return temp;
}
```

```
/* These variables are undefined just in case they are used in a different module of the
 *  main routine.  The undefined has been added to make the code portable. */

#undef M1
#undef IA1
#undef IC1
#undef RM1
#undef M2
#undef IA2
#undef IC2
#undef RM2
#undef M3
#undef IA3
#undef IC3
```

APPENDIX B

```
/***********************************************************************
* "White" noise tester using FFT method
* J. Ionata, 11-AUG-1992
* A. Edmonds 14-SEP-1992
* Based on TR 6843 by Al Nuttall, 01-DEC-1982
***********************************************************************/
#define K 1048576
#define M (2 * K)
#define thrshold 1.5
#define trials 1000
#define SEED 13
#include <stdio.h>
#include <math.h>
main ()
{
    double x_rl[M];                 /* uniform random data */
    double x_im[M];
    double ran1();
    double R_hat;                   /* correlation estimate */
    double err, prob15;          /* error measure "E" */
    double white, avg, wtotal;   /* whiteness measure */
    double leastw, mostw;           /* variances */
    double mag_sqrd;                /* magnitude squared value */
    int n;                                /* delay (index) */
    int k, m, abovethrs;            /* thresholds */
    int t;                          /* t number of trials */

    /* begin main program */
    m = M;
    k = K;
    abovethrs = 0.0;
    leastw = 100.0;
    mostw = 0.0;

    /* printf ("\nClearing arrays ... "); */
    for (n = 0; n < m; n++)
    {
        x_rl[n] = 0.0;
        x_im[n] = 0.0;
    }
    /* printf("done."); */

    for (t = 0; t < trials; t++)
    {
    err = 0.0;
    R_hat = 0.0;
    for (n = k; n < m; n++)
    {
        x_rl[n] = 0.0;
        x_im[n] = 0.0;
    }
    /* printf ("\nCalculating uniform random numbers ... "); */
    for (n = 0; n < k; n++)
    {
```

```
    /*  Candidate Random number generator routine ran1, rand &
        random */
          x_rl[n] = ran1(SEED)-0.5;           /* random -0.5 to 0.5 */
          x_im[n] = 0.0;
    }
    /* printf("done."); */

    /* printf("\nDoing FFT ... "); */
    anfft(m, x_rl, x_im);
    /* printf("done."); */

    /* printf("\nDoing summation ... "); */
    for (n = 0; n < m; n++)
    {
          mag_sqrd = x_rl[n]*x_rl[n] + x_im[n]*x_im[n];
          R_hat += mag_sqrd;
          err += mag_sqrd * mag_sqrd;
    }
    /* printf("done."); */

    white = (m * err / (R_hat * R_hat)) -1;
    wtotal += white;
    if (white > thrshold)
    abovethrs++;

    if (white > mostw)              /* highest whiteness */
    mostw = white;

    if (white < leastw)             /* lowest whiteness */
    leastw = white;
    }
    avg = wtotal/(trials);          /* average whiteness */

    printf ("\naverage whiteness = %e", avg);
    printf ("\nminimum whiteness = %e", leastw);
    printf ("\nmaximum whiteness = %e", mostw);
    printf ("\nnumber of trials = %4d", trials);

    prob15 = (float)abovethrs/trials;
    printf ("\nProbability Q > 1.5 = %e", prob15);
    printf ("\nProgram done.\n");
}
```

BIBLIOGRAPHY

Stochastic Processes

Doob, J. L., "Stochastic Processes," New York, 1953 (Wiley Classics Library Edition Published 1990.)

H. Cramer, "Random Variables and Probability Distributions," University Press, Cambridge, 1970.

Probability Distributions

Heyman, D.P. and Sobel, M.J., eds., "Stochastic Models, Vol. 2 of Handbooks in Operations Research and Management Science," Elsevier Science Publishers B.V., North Holland, 1990.

Clarke, A.B. and Disney, R.L., "Probability and Random Processes for Engineers and Scientists," John Wiley & Sons, New York, 1970.

Methods of Testing Algorithms

Knuth, D., "Seminumerical Algorithms, Vol. 2 of The Art of Computing Programming," Addison-Wesley, Reading, MA 1981.

Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling,W.T., "Numerical Recipes: The Art of Scientific Computing," University Press, Cambridge, 1970.

Time and Frequency Domain Applications

Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling,W.T., "Numerical Recipes: The Art of Scientific Computing," University Press, Cambridge, 1970.

Algorithms

Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling,W.T., "Numerical Recipes: The Art of Scientific Computing," University Press, Cambridge, 1970.

Golomb, S.W., "Shift Register Sequences," Holden-Day Inc., San Francisco, 1967.

DISTRIBUTION LIST

Internal

Codes:	10	
	21	
	215	
	2152	(J. Sanchis)
	2153	(H. Watt, J. Munoz, J. Ionata, A. Edmonds (2))
	302	(A. Nuttall)
	0261	(NLON Library (2))
	0262	(NPT Library)

Total: 13