An Inverse Method for The Generation of Random Normal Deviates on Large-Scale Computers

By Mervin E. Muller

1. Introduction. Many applications of electronic computers require efficient generation of large numbers of random normal deviates; see, e.g. [7], [12], [14], [17]. Tables of random normal deviates are of course available, for example [18], [23], but they are not sufficiently extensive for many purposes and an outside source of this kind cannot usually be used effectively by the computer. What is required is some method of generation which can be rapidly carried out by the machine itself.

Methods are available by which pseudo random numbers may be produced, see for example [2], [5], [6], [10], [11], [13], [20], [22]. Judging by the results given, for example, by [5], [6], [20], [21], the most satisfactory procedure now in use for generating random numbers is that based on "residue class" techniques, see for example [13], [20]. We shall not consider here the validity of these methods but shall assume that some satisfactory method of producing uniform random deviates is available from which random normal deviates are to be produced. Several ways of generating random normal deviates are known; see, for example, [1], [6], [21], but as is shown in [15], these methods are not as fast nor as reliable, especially for extreme tail values, as that developed here.

In this paper the inverse of the Normal Distribution Function is approximated stepwise by rational functions. Hence a uniform random deviate may be transformed into a normal deviate. The details of the approximations have been obtained for use on a large scale binary machine with index registers, where it is possible to utilize memory space in order to obtain greater speed.

The present approach can be used, if desired, in the course of generating arbitrary n-variate normal deviates or Chi-squared variates.

2. Inverse method. In principle, the inverse method of generating a normal deviate x from a uniform deviate u is well known. The problem is to find the inverse relationship $x = x(u)$ given that

$$u = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt.$$

The actual determination of $x(u)$ offers certain numerical difficulties when it is desired to generate reliable normal deviates, especially for large values of x. We have considered this problem for two aspects. First, speed and secondly and equally important, reliability. We have insisted that the maximum absolute error in x should be less than 4×10^{-4} in the range $-5 \leq x \leq 5$, where we assume

Received October 26, 1957. Prepared in connection with research sponsored by the Office of Ordnance Research, U. S. Army; Statistical Techniques Research Group, Princeton University, Contract No. DA 36-034-ORD 2297.

u is correct, and where Prob $\{-5 \leq X \leq 5\} > 1 - 6 \times 10^{-7}$ so that x is correct to within 4×10^{-4} except for an event of probability less than 6×10^{-7}.

The relation $x = x(u)$ is approached in stages. The interval $[0, 1]$ for u is subdivided so that over each sub-interval it is possible to obtain a reliable and fast procedure for computing x. Over most of $(0, 1)$ $x = x(u)$ is approximated by Chebyshev polynomials. As x becomes large in absolute value it is necessary to increase the degree of the approximating polynomial. However, even though the degree of the polynomial increases, the frequency with which these approximations are needed decreases; hence this method will use, on the average, a low order approximating polynomial for $x = x(u)$. Due to symmetry, it is actually only necessary to study $x = x(u)$ for $1/2 \leq u \leq 1$.

For speed in computing it is best to have the Chebyshev polynomials of as low degree as possible, subject to the specified level of accuracy. Following this approach one could find sub-intervals of greatest or optimum width, see [3]. Two drawbacks then become apparent. If on a binary computer the sub-interval widths are not some negative power of two, considerable computing time is spent in mapping a given sub-interval onto the range $(-1, 1)$ which is needed when using a Chebyshev approximation. More important, if the intervals are not of uniform width and a negative power of two, considerable machine time, and memory space, is required to select the appropriate approximation. For, it will not then be possible to use an index register to select the appropriate approximation. Taking these facts into consideration, and taking into account memory space requirements, it was decided to have the widths of the sub-intervals equal $1/128$. With this choice of interval width the largest part of $(0, 1)$, namely $1/8 \leq u \leq 7/8$, could be approximated by linear functions. Quadratic and quartic approximations are used for $7/8 < u \leq 127/128$. For $127/128 < u < 1$, and by symmetry $0 < u < 1/128$, $x = x(u)$ has a singularity of logarithmic type; consequently, for u in this sub-interval an approximation of more subtle type than Chebyshev polynomials is needed.

3. Use of polynomials. Since the techniques used in approximating $x = x(u)$ have possible application when considering the generation of other possible pseudo random variables we shall review the techniques in detail.

To approximate $x = x(u)$ in the jth sub-interval,

$$\left(\frac{64 + j - 1}{128} \leq u \leq \frac{64 + j}{128}\right), \quad j = 1(1)63,$$

by Chebyshev polynomials we shall make use of the technique of Interpolating Chebyshev Polynomials as developed by C. Lanczos, see for example [8], [9]. To begin, the jth sub-interval is transformed onto $(-1, 1)$ by using

$$r = 256u - 127 - 2j.$$

Let $r = \cos\theta$, $0 \leq \theta \leq \pi$. Let $T_i(r) = \cos[i(\cos^{-1} r)]$ denote the ith Chebyshev polynomial. Thus for the jth sub-interval, $x = x_j(r)$ will be approximate by a polynomial of nth degree, namely,

$$x_{nj}(r) = 1/2\, d_{0j} + \sum_{i=1}^{n} d_{ij} T_i(r);$$

where the d_{ij}'s are obtained from

$$d_{ij} = \frac{2}{n+1} \sum_{k=0}^{n} x(r_k) T_i(r_k),$$

and where the r_k's are the zeros of the $(n+1)$th Chebyshev polynomial. Thus $r_k = \cos(\pi(2k+1)/2(n+1))$. This approach approximately minimizes the maximum error throughout the sub-interval.

For computing purpose it is more convenient to express the $T_i(r)$'s as polynomials in r, see Lanczos [8], [9], for example $T_1(r) = r$, $T_2(r) = 2r^2 - 1$. One then obtains

$$x_{nj}(r) = \sum_{i=0}^{n} a_{ij} r^i. \tag{1}$$

The coefficients a_{ij} are given in section 4.

In calculating the d_{ij}'s, and hence the a_{ij}'s, it is necessary to evaluate $x = x(u)$ for u_{kj}'s where

$$u_{kj} = \frac{r_k + 127 + 2j}{256},$$

and this was done for the seven intervals $j = 57(1)63$, i.e., $7/8 < u \leq 127/128$. The values of $x(u_{kj})$ were obtained by cubic inverse interpolation in the National Bureau of Standards Tables of the Normal Probability Function, [19].

However, for the intervals, $j = 1(1)56$, i.e., $1/2 \leq u \leq 7/8$, where the linear approximations are adequate the essential computations for the a_{ij}'s are concerned with finding the inverse values, namely $x = x(u_{kj})$. However, for the linear cases the a_{ij}'s given in section 4 were not obtained in the above manner since we had previously needed to have values of x corresponding to $u_s = (128 + s)/256$, $s = 0, 1, 2, \cdots, 128$. Consequently we avoided the additional labor of finding the necessary x's by utilizing the available $x(u_s)$'s. These values are included as Table 5 in the Appendix. Thus, while not fitting the straight lines by first degree Chebyshev polynomials we were able to obtain the desired level of precision as follows: In the jth interval, where it is appropriate to fit a straight line, consider u_j, u_{j+1}, and $u_{j,1/2} = (u_j + u_{j+1})/2$. Let x_j, x_{j+1}, $x_{j,1/2}$ denote the corresponding x values. The jth line is fitted such that the deviations at the ends of the class interval are equal and such that the deviation at the mid-point, namely $u_{j,1/2}$, is equal in magnitude to the deviations at the end points, but is of opposite sign. This approach gives essentially a Chebyshev-type approximation in that we are striving to minimize the maximum error in a given sub-interval. By straightforward computations one then obtains that

$$a_{0j} = \frac{2(x_{j+1} + x_{j,1/2}) - (x_{j+1} - x_j)(255 + 4j)}{4},$$

$$a_{ij} = 128(x_{j+1} - x_j).$$

In the selection of the width of a sub-interval, w_j, or for determining the appropriate degree of the Chebyshev Polynomial for a given sub-interval, the following

result is used, see [3] for more general details,

$$w_j = 4\left[\frac{(n+1)!}{2}\,\epsilon\,|x^{(n+1)}(c_j)|^{-1}\right]^{\frac{1}{n+1}},$$

where ϵ is the largest absolute error that is tolerable, in our case $\epsilon = 4 \times 10^{-4}$, and where $x^{(n+1)}(u)$ is the $(n+1)$st derivative of $x = x(u)$ with respect to u.

4. Table of coefficients. The following tables provide the necessary coefficients for the approximations given by equation (1) of section 3.

TABLE 1. *Linear Cases*

j	a_{0j}	a_{1j}	j	a_{0j}	a_{1j}
1	−1.25339 449	2.50678 851	29	−1.56607 868	2.98461 325
2	−1.25388 344	2.50775 044	30	−1.59698 101	3.02714 356
3	−1.25487 723	2.50967 688	31	−1.63039 945	3.07264 746
4	−1.25639 365	2.51257 300	32	−1.66656 010	3.12136 699
5	−1.25845 203	2.51644 669	33	−1.70571 765	3.17357 468
6	−1.26107 329	2.52130 843	34	−1.74815 989	3.22957 838
7	−1.26428 016	2.52717 152	35	−1.79421 327	3.28972 698
8	−1.26809 727	2.53405 216	36	−1.84424 956	3.35441 736
9	−1.27255 126	2.54196 946	37	−1.89869 401	3.42410 298
10	−1.27767 105	2.55094 571	38	−1.95803 516	3.49930 401
11	−1.28348 794	2.56100 644	39	−2.02283 720	3.58062 028
12	−1.29003 594	2.57218 075	40	−2.09375 508	3.66874 687
13	−1.29735 187	2.58450 135	41	−2.17155 347	3.76449 389
14	−1.30547 572	2.59800 495	42	−2.25713 085	3.86881 137
15	−1.31445 096	2.61273 256	43	−2.35154 997	3.98282 083
16	−1.32432 485	2.62872 974	44	−2.45607 734	4.10785 604
17	−1.33514 882	2.64604 704	45	−2.57223 456	4.24551 603
18	−1.34698 984	2.66475 763	46	−2.70077 415	4.39645 556
19	−1.35986 540	2.68485 495	47	−2.84833 659	4.56815 521
20	−1.37390 816	2.70651 014	48	−3.01223 276	4.75713 372
21	−1.38914 726	2.72973 042	49	−3.19590 941	4.96703 562
22	−1.40567 396	2.75461 642	50	−3.40829 673	5.20760 458
23	−1.42357 635	2.78126 053	51	−3.65403 652	5.48350 516
24	−1.44295 134	2.80976 489	52	−3.93953 404	5.80124 828
25	−1.46390 578	2.84024 264	53	−4.27256 248	6.16869 364
26	−1.48655 769	2.87281 914	54	−4.67044 630	6.60394 516
27	−1.51103 773	2.90763 359	55	−5.15058 375	7.12472 205
28	−1.53749 093	2.94484 078	56	−5.74560 472	7.76467 397

TABLE 2. *Quadratic Cases*

j	a_{0j}	a_{1j}	a_{2j}
57	1.56668 859	−0.03343 48405	.00087 5575
58	1.63732 538	−0.03745 15701	.00114 804
59	1.71722 812	−0.04284 26652	.00157 546
60	1.80989 233	−0.05049 00254	.00230 549
61	1.92135 077	−0.06226 30013	.00372 027
62	2.06352 790	−0.08299 31005	.00708 977

TABLE 3. *Quartic Case*

j	a_{0j}	a_{1j}	a_{2j}	a_{3j}	a_{4j}
63	+2.26622 681	+0.12757 931	+0.01844 432	+0.00424 42872	+0.00104 06032

5. The interval $(127/128 < u < 1)$. In view of the fact that $x(u)$ may be looked upon as a function having a singularity of logarithmic type in this interval, it is necessary to abandon the use of a polynomial type approximation here. A satisfactory rational approximation is obtained by using a truncated continued fraction expansion, see for example [4], [16]. For any given value of u the corresponding value of x is approximated by the following recurrence relation. The kth convergent to $x(u)$ is given by

$$x_k(u) = \frac{M_k(u)}{N_k(u)},$$

where $M_k(u)$ and $N_k(u)$ are determined as follows:

$$(2) \quad M_{k+1}(u) = d_k M_k(u) + (u - u_{k-1})M_{k-1}(u); \quad M_0(u) = 1; \quad M_1(u) = d_0;$$

$$(3) \quad N_{k+1}(u) = d_k N_k(u) + (u - u_{k-1})N_{k-1}(u); \quad N_0(u) = 0; \quad N_1(u) = 1;$$

and where d_k denotes a selected value of the kth inverted divided difference of $x(u)$.

In order to insure sufficient precision it is suggested that the machine program test to see if $u > .99999$. When u is greater than .99999 it is suggested that the additional significant figures for u_k, $k = 10(1)14$, be utilized. Double precision operations are not necessary if these u_k are stored with the first three nines suppressed, u must then be shifted the appropriate number of places before performing $u - u_k$.

Following [4], page 406, $x_k(u)$ can be computed from

$$(4) \quad x_k(u) = d_0 + \sum_{n=1}^{k} \frac{(-1)^{n+1}(u - u_0)(u - u_1) \cdots (u - u_{n-1})}{N_n(u)N_{n+1}(u)}.$$

To insure that $x(u)$ can be approximated to within 4×10^{-4} it is necessary to stop at an appropriate value of $k = k(u)$ since the approximation being employed is essentially a divergent expansion. Numerical evaluations verified that it is sufficient to have the machine program terminate as soon as one of the following three conditions is satisfied, namely: (i) select $x_k(u)$ as the approximation to $x(u)$

TABLE 4. *Constants for Rational Approximation*

k	u_k	d_k
0	.99223 97464	2.42000 0000
1	.99461 38540	0.01826 2366
2	.99653 30262	−0.65518 1080
3	.99813 41867	0.02399 7757
4	.99903 23968	−0.26737 1460
5	.99931 28620	0.01654 1263
6	.99966 30707	−0.14194 9840
7	.99984 08914	0.00997 32778
8	.99992 76519	−0.06004 9158
9	.99996 83287	0.00511 81541
10	.99998 66542 51	−0.02329 9296
11	.99999 45874 56	0.00241 07770
12	.99999 78875 45	−0.00863 34192
13	.99999 92066 72	0.00100 76316
14	.99999 97133 484	−0.00308 28145

if $r_k = |x_k(u) - x_{k-1}(u)| < 4 \times 10^{-4}$, (ii) select $x_k(u)$ as the approximation to $x(u)$ if $r_k - r_{k+1} < 0$, and (iii) if $k = 14$ terminate the process and select $x_{14}(u)$ as the approximation to $x(u)$.

The value of k increases as u increases in the interval $(127/128 < u < 1 - 3 \times 10^{-7})$, and consequently so does the computing time. However, this subinterval will, on the average, increase the necessary computing time very little. It should be kept in mind that while this method is very appropriate and convenient for a machine which performs "floating point" multiplication, this approach would create serious scaling difficulties for a "fixed point" mode of operation.

6. Concluding comments. A detailed comparison of this method with other known methods is given in [15]. The technique proposed in this paper yields accuracy comparable with, or better than, most previous proposals using about one-quarter the computing time while requiring about twice as many memory locations.

The author wishes to express his appreciation to Dr. G. E. P. Box, Dr. Collin Mallows, and Dr. John W. Tukey for their interest and generous comments concerning this work. The author is also indebted to Mrs. A. Schay for her excellent help with the numerical computations.

Princeton University
Princeton, New Jersey

1. G. E. P. Box & M. E. Muller, "A note on the generation of normal deviates," *Ann. Math. Stat.*, to be published.
2. George E. Forsythe, "Generation and testing of random digits at the National Bureau of Standards, Los Angeles," *Monte Carlo Method*, NBS, Applied Mathematics Series 12, U. S. Government Printing Office, Washington, D. C., 1951, p. 34–35. [*MTAC*, Rev. **42**, v. XI, 1957, p. 43–44.]
3. Joseph O. Harrison, Jr., "Piecewise polynomial approximations for large-scale digital calculations." [*MTAC*, v. III, 1949, p. 400–407.]
4. F. B. Hildebrand, *Introduction to Numerical Analysis*, McGraw-Hill Book Co., Inc., New York, 1956.
5. D. L. Johnson, "Generating and testing pseudo random numbers on the IBM type 701," *MTAC*, v. X, 1956, p. 8–13.
6. M. L. Juncosa, "Random number generation on the BRL High-Speed computing machines," Report 855, Ballistic Research Laboratories, Aberdeen Proving Ground, Maryland, 1953.
7. Herman Kahn, "Applications of Monte Carlo," RAND Project Report, 1954, Revised 1956, RAND Project Report RM-1237-AEC.
8. C. Lanczos, "Trigonometric interpolation of empirical and analytical functions," *Jn. Math. Phys.*, v. 17, 1938, p. 123–199.
9. C. Lanczos, *Tables of Chebyshev Polynomials* $S_n(x)$ and $C_n(x)$, NBS, Applied Mathematics Series 9, U. S. Government Printing Office, Washington, D. C., 1952. [*MTAC*, Rev. **1103**, v. 7, 1953, p. 174.]
10. D. H. Lehmer, "Mathematical methods in large scale computing units," *Proceedings Second Symposium on Large-Scale Digital Calculating Machinery*, 1949, p. 141–146. Harvard University Press, Cambridge, Mass., 1951.
11. D. H. Lehmer, Description of "Random number generation of the BRL high-speed computing machines," *Mathematical Reviews*, v. 15, 1954, p. 559.
12. NBS, Applied Mathematics Series 12, Monte Carlo Methods, 1951, p. 34–35.
13. J. Moshman, "The generation of pseudo-random numbers on a decimal calculator," *Jn.*, Assoc. for Computing Machinery, v. 1, 1954, p. 88–91.
14. M. E. Muller, "Some continuous Monte Carlo methods for the Dirichlet problem," *Ann. Math. Stat.*, v. 27, p. 569–589.
15. M. E. Muller, "A comparison of methods for generating normal deviates," Technical Report No. 9, Statistical Techniques Research Group, Department of Mathematics, Princeton University, to be published.
16. N. E. Norlund, *Vorlesungen Uber Differenzenrechnung*, Springer, Berlin, 1924.
17. Herbert A. Meyer, Editor, *Symposium on Monte Corlo Methods*, held at the University of Florida, 1954, John Wiley and Sons, Inc., New York, 1956. [*MTAC*, Rev. **43**, v. XI, 1957, p. 44–46.]
18. The Rand Corporation, *One Million Random Digits and 100,000 Normal Deviates*, The Free Press, Glencoe, Illinois, 1955. [*MTAC*, Rev. 11, v. X, 1956, p. 39–43.]

19. NBS, Applied Mathematics Series 23, *Tables of Normal Probability Functions*, U. S. Government Printing Office, Washington, D. C., 1953.
20. O. TAUSSKY & J. TODD, "Generation and testing of pseudo-random numbers," *Symposium on Monte Carlo Methods*, University of Florida, 1954, John Wiley and Sons, Inc., 1956, p. 15–28.
21. D. TEICHROEW, *Distribution Sampling with High-Speed Computers*, Ph.D. Thesis, University of North Carolina, 1953.
22. D. F. VOTAW, JR. & J. A. RAFFERTY, "High speed sampling," *MTAC*, v. 5, 1951, p. 1–8.
23. H. WOLD, "Random normal deviates," *Tracts for Computers*, No. XXV, Cambridge University Press, New York, 1948.

Appendix

This table gives values of $x(u)$ corresponding to the Normal Distribution for $F(x_j) = 1/2 + j/256$, $j = 1(1)127$. The values were obtained from the National Bureau of Standard Tables of the Normal Probability Function, [19], by using the inversion formula:

$$x(u) = u_0 + \frac{a}{2Q_0} + \frac{u_0}{2}\left(\frac{a}{2Q_0}\right)^2 + \frac{(2u_0^2 + 1)}{6}\left(\frac{a}{2Q_0}\right)^3,$$

where u_0 is the nearest tabulated entry to u, and $u - u_0 = a$, and where

$$Q_0 = \frac{1}{\sqrt{2\pi}} e^{-[x(u_0)]^2/2}.$$

TABLE 5. *Inverse Values for the Normal Distribution*

j	$F(x_j)=1/2+j/256$	x_j	j	$F(x_j)=1/2+j/256$	x_j
1	0.50390 625	0.00979 167	33	0.62890 625	0.32895 791
2	0.50781 250	0.01958 429	34	0.63281 250	0.33931 161
3	0.51171 875	0.02937 878	35	0.63671 875	0.34970 180
4	0.51562 500	0.03917 609	36	0.64062 500	0.36013 003
5	0.51953 125	0.04897 716	37	0.64453 125	0.37059 729
6	0.52343 750	0.05878 294	38	0.64843 750	0.38110 545
7	0.52734 375	0.06859 437	39	0.65234 375	0.39165 587
8	0.53125 000	0.07841 241	40	0.65625 000	0.40225 007
9	0.53515 625	0.08823 802	41	0.66015 625	0.41288 960
10	0.53906 250	0.09807 215	42	0.66406 250	0.42357 608
11	0.54296 875	0.10791 578	43	0.66796 875	0.43431 116
12	0.54687 500	0.11776 987	44	0.67187 500	0.44509 652
13	0.55078 125	0.12763 542	45	0.67578 125	0.45593 392
14	0.55468 750	0.13751 340	46	0.67968 750	0.46682 512
15	0.55859 375	0.14740 482	47	0.68359 375	0.47777 199
16	0.56250 000	0.15731 068	48	0.68750 000	0.48877 641
17	0.56640 625	0.16723 201	49	0.69140 625	0.49984 034
18	0.57031 250	0.17716 982	50	0.69531 250	0.51096 581
19	0.57421 875	0.18712 516	51	0.69921 875	0.52215 488
20	0.57812 500	0.19709 908	52	0.70312 500	0.53340 971
21	0.58203 125	0.20709 265	53	0.70703 125	0.54473 251
22	0.58593 750	0.21710 695	54	0.71093 750	0.55612 559
23	0.58984 375	0.22714 306	55	0.71484 375	0.56759 132
24	0.59375 000	0.23720 211	56	0.71875 000	0.57913 216
25	0.59765 625	0.24728 522	57	0.72265 625	0.59075 066
26	0.60156 250	0.25739 353	58	0.72656 250	0.60244 945
27	0.60546 875	0.26752 821	59	0.73046 875	0.61423 129
28	0.60937 500	0.27769 044	60	0.73437 500	0.62609 901
29	0.61328 125	0.28788 143	61	0.73828 125	0.63805 558
30	0.61718 750	0.29810 241	62	0.74218 750	0.65010 407
31	0.62109 375	0.30835 463	63	0.74609 375	0.66224 768
32	0.62500 000	0.31863 936	64	0.75000 000	0.67448 975

TABLE 5—*Continued*

j	$F(x_j)=1/2+j/256$	x_j	j	$F(x_j)=1/2+j/256$	x_j
65	0.75390 625	0.68683 375	97	0.87890 625	1.16953 661
66	0.75781 250	0.69928 330	98	0.88281 250	1.18916 435
67	0.76171 875	0.71184 220	99	0.88671 875	1.20926 123
68	0.76562 500	0.72451 438	100	0.89062 500	1.22984 876
69	0.76953 125	0.73730 400	101	0.89453 125	1.25099 172
70	0.77343 750	0.75021 538	102	0.89843 750	1.27268 865
71	0.77734 375	0.76325 304	103	0.90234 375	1.29502 241
72	0.78125 000	0.77642 176	104	0.90625 000	1.31801 090
73	0.78515 625	0.78972 652	105	0.91015 625	1.34171 784
74	0.78906 250	0.80317 257	106	0.91406 250	1.36620 382
75	0.79296 875	0.81676 542	107	0.91796 875	1.39153 749
76	0.79687 500	0.83051 088	108	0.92187 500	1.41779 714
77	0.80078 125	0.84441 508	109	0.92578 125	1.44507 258
78	0.80468 750	0.85848 447	110	0.92968 750	1.47345 903
79	0.80859 375	0.87272 589	111	0.93359 375	1.50310 294
80	0.81250 000	0.88714 656	112	0.93750 000	1.53412 054
81	0.81640 625	0.90175 411	113	0.94140 625	1.56668 859
82	0.82031 250	0.91655 667	114	0.94531 250	1.60100 866
83	0.82421 875	0.93156 283	115	0.94921 875	1.63732 538
84	0.82812 500	0.94678 176	116	0.95312 500	1.67594 192
85	0.83203 125	0.96222 320	117	0.95703 125	1.71722 812
86	0.83593 750	0.97789 754	118	0.96093 750	1.76167 041
87	0.83984 375	0.99381 591	119	0.96484 375	1.80989 233
88	0.84375 000	1.00999 017	120	0.96875 000	1.86273 187
89	0.84765 625	1.02643 306	121	0.97265 625	1.92135 077
90	0.85156 250	1.04315 826	122	0.97656 250	1.98742 789
91	0.85546 875	1.06018 048	123	0.98046 875	2.06352 790
92	0.85937 500	1.07750 557	124	0.98437 500	2.15387 469
93	0.86328 125	1.09518 065	125	0.98828 125	2.26622 681
94	0.86718 750	1.11319 428	126	0.99218 750	2.41755 902
95	0.87109 375	1.13157 656	127	0.99609 375	2.66006 747
96	0.87500 000	1.15035 938			

Algebraic Approximations for Laplace's Equation in the Neighborhood of Interfaces

By J. W. Sheldon

1. Introduction. Let it be required to solve the following problem:

Problem A:

Let C_i, $i = 1, 2$, be two simple, closed plane curves with continuous curvature. Let C_2 enclose all the points of C_1. Let G_1 be the region interior to C_1. Let G_2 be the region interior to C_2 and exterior to C_1. Let W be a continuous, bounded function of position on C_2. Let it be required to find harmonic functions $V^{(i)}$ such that

(a) $V^{(2)} = W$ on C_2.

(b) $V^{(i)}$ is regular and bounded in G_i.

Received September 19, 1957.