INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION

INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
34100 TRIESTE (ITALY) - P.O.B. 586 - MIRAMARE - STRADA COSTIERA 11 - TELEPHONE: 2240-1
CABLE: CENTRATOM - TELEX 460392 - I

SECOND SCHOOL ON ADVANCED TECHNIQUES
IN COMPUTATIONAL PHYSICS
(18 January - 12 February 1988)

SMR.282/ 10

M.C.D. GENERATORS

F.JAMES
F.BERMEJO

EXERCISES

M.C.D. GENERATORS

F. James

F. Bermejo

1.) STUDY OF R.N. GENERATORS

a.) The mid-squares method, proposed by J. von Neuman was apparently the first algorithm proposed for the generation of psuedorandom numbers, and proceeds as follows:

Suppose we wish to generate four-digit integers and the last number generated was ABCD. Let's assume that we are using 8-digit decimals and that in successive steps, we just take the four middle digits as the number we want. To obtain the next number in the sequence, we square the last one and use the middle four digits of the product, i.e.

$$\underbrace{XYABCDZW}_{\text{squared}} = \underbrace{RWEFGDYV}_{\text{squared}} = \underbrace{SVHLMOTP}_{\text{squared}} \text{ (etc.)}$$

Study the statistical quality of this RNG by means of:

a-1) Its dependence upon the "seed" (the first number).

a-2) The repetition cycle (How long before the sequence ABCD is repeated.

b.) The Fibonacci method consists in adding two preceeding numbers and taking the remainder when the sum is taken with respect to some modulus i.e.:

$$X_i = (X_{i-1} + X_{i-2}) \bmod m$$

b-1) Write a code to study the quality of this RNG.

b-2) Consider the recursive formula,

$$X_i = (a X_{i-1} + c) \bmod m$$

where a, c and m are parameters which depend upon the particular word-length and determine the statistical quality of the generator. For c=o, the method is called a "pure multiplicative congruential" generator. Write a code to generate RN's using this method. (Hint: use a modulus $m = 2^{15} - 1$). Test the quality of the distribution by means of the X^2 statistic. (Hint: Partition the outcomes in m-subsets: $X^2 <$ 0.10, $0.10 < X^2 < 0.2$ etc. and compute the number of occurences fi. If we denote the expected frequency as fi, the computed quantity,

$$X^2 = \sum_{i=1}^{\mu} \frac{(fi - fi)^2}{fi}$$

should approach the X^2 - distribution for m-1 degrees of freedom. The approximation tends to be reasonable for large numbers of cells 300).

c.) The combination of two pseudorandom sequences X and Y to produce a thrid Z tends to reduce nonrandomness. Code the algorithms:

a.) set $Z = (X + Y) \bmod m$

b.) use $X = 171\,X \bmod 30269$

$Y = 172\,Y \bmod 30307$

and compare its quality with that of the preceeding one by X - testing.

2.) Make a Gaussian RNG by means of the central-limit theorems in the following way; consider a set of $V_1, \ldots, V_n$ random variables and set $N = \sum_i V_i$. As $n \to \infty$ N, tends to be normally-distributed. Since we have to settle for a finite n, a convenient way to obtain N is:

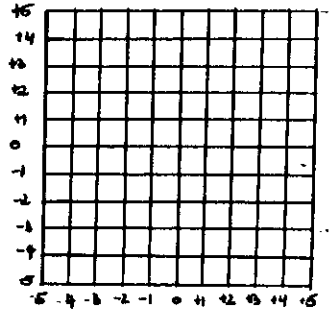
$$N = \sum_{i=1}^{12} V_i - 6$$

where the V_i are uniformly-generated psuedorandom numbers. Code the above equation and compute the expectation value and variance of the obtained distribution.

3.) Study the properties of the distribution obtained by dividing two independent Gaussian-distributed variables X, Y, (i.e. Z = X/Y). Compute the expectation and variance of Z numerically. Take X and Y = N(1,1).

4.) 2D RANDOM WALK

Consider a planar square lattice like:



Suppose you are initially located at the lattice point (0,0). A "random walk" is simulated if at each lattice point a random choice is made about the direction of your next step.

a.) Code an algorithm to perform such a walk starting from an arbitary point.

b.) Suppose that every time you reach the boundaries you get reflected (unable to leave the lattice) unless you are (-2,5) or (4,5). Compute the number of steps required to exit (try at least 10 simulations).

c.) Self-avoiding-walk.

Suppose that at each lattice point the random choice is subject to the restrictions that:

a.) you will never visit a lattice point more than once.

b.) all the boundaries are closed.

Write a program to simulate such a movement, bearing in mind that the simulation will stop by either achieving a maximum number of steps (200) or by getting stuck into a point (unable to move without violating the constraints a & b).

5.) BRUTE FORCE INTEGRATION.

Compute the value of the definite integral,

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

by means of a program which determines the area under the function f(x) = 2 . The way tp proceed would be to generate random numbers and check if the associated point lays below f(x) (i.e. P is below f(x) $0 < X < 1$) Generate at least 5000 events $0 < Y < 2$).

6.) Code the Adaptive integration algorithm given in the lectures (use the scheme, the abscissas and weights given separately).

ADAPTIVE QUADRATURE.

A GOOD ALGORITHM:

R_{65}

ON ONE INTERVAL, USE R_{65} :

G_n = GAUSS RULE with n points

$I = G_6$

$\Delta = |G_6 - G_5|$

THEN:

1. APPLY R_{65} TO TOTAL INTERVAL
2. IF Δ TOO BIG GO TO 3 ELSE FINISHED
3. DIVIDE IN HALVES THE INTERVAL WITH THE LARGEST Δ
4. APPLY R_{65} TO BOTH HALVES
5. SET $I = \sum_{\text{ALL INTERVALS}} I_i$

$\Delta = \sum \Delta_i$

6. GO TO 2

Table 25.4 ABSCISSAS AND WEIGHT FACTORS FOR GAUSSIAN INTEGRATION

$$\int_{-1}^{+1} f(x)dx = \sum_{i=1}^{n} w_i f(x_i)$$

Abscissas $= \pm x_i$ (Zeros of Legendre Polynomials) Weight Factors $= w_i$

$\pm x_i$	w_i
$n = 2$	
0.57735 02691 89626	1.00000 00000 00000
$n = 3$	
0.00000 00000 00000	0.88888 88888 88889
0.77459 66692 41483	0.55555 55555 55556
$n = 4$	
0.33998 10435 84856	0.65214 51548 62546
0.86113 63115 94053	0.34785 48451 37454
$n = 5$	
0.00000 00000 00000	0.56888 88888 88889
0.53846 93101 05683	0.47862 86704 99366
0.90617 98459 38664	0.23692 68850 56189
$n = 6$	
0.23861 91860 83197	0.46791 39345 72691
0.66120 93864 66265	0.36076 15730 48139
0.93246 95142 03152	0.17132 44923 79170
$n = 7$	
0.00000 00000 00000	0.41795 91836 73469
0.40584 51513 77397	0.38183 00505 05119
0.74153 11855 99394	0.27970 53914 89277
0.94910 79123 42759	0.12948 49661 68870
$n = 8$	
0.18343 46424 95650	0.36268 37833 78362
0.52553 24099 16329	0.31370 66458 77887
0.79666 64774 13627	0.22238 10344 53374
0.96028 98564 97536	0.10122 85362 90376
$n = 9$	
0.00000 00000 00000	0.33023 93550 01260
0.32425 34234 03809	0.31234 70770 40003
0.61337 14327 00590	0.26061 06964 02935
0.83603 11073 26636	0.18064 81606 94857
0.96816 02395 07626	0.08127 43883 61574
$n = 10$	
0.14887 43389 81631	0.29552 42247 14753
0.43339 53941 29247	0.26926 67193 09996
0.67940 95682 99024	0.21908 63625 15982
0.86506 33666 88985	0.14945 13491 50581
0.97390 65285 17172	0.06667 13443 08688
$n = 12$	
0.12523 34085 11469	0.24914 70458 13403
0.36783 14989 98180	0.23349 25365 38355
0.58731 79542 86617	0.20316 74267 23066
0.76990 26741 94305	0.16007 83285 43346
0.90411 72563 70475	0.10693 93259 95318
0.98156 06342 46719	0.04717 53363 86512

$\pm x_i$	w_i
$n = 16$	
0.09501 25098 37637 440185	0.18945 06104 55068 496285
0.28160 35507 79258 913230	0.18260 34150 44923 588867
0.45801 67776 57227 386342	0.16915 65193 95002 538189
0.61787 62444 02643 748447	0.14959 59888 16576 732081
0.75540 44083 55003 033895	0.12462 89712 55533 872052
0.86563 12023 87831 743880	0.09515 85116 82492 784810
0.94457 50230 73232 576078	0.06225 35239 38647 892863
0.98940 09349 91649 932596	0.02715 24594 11754 094852
$n = 20$	
0.07652 65211 33497 333755	0.15275 33871 30725 850698
0.22778 58511 41645 078080	0.14917 29864 72603 746788
0.37370 60887 15419 560673	0.14209 61093 18382 051329
0.51086 70019 50827 098004	0.13168 86384 49176 626898
0.63605 36807 26515 025453	0.11819 45319 61518 417312
0.74633 19064 60150 792614	0.10193 01198 17240 435037
0.83911 69718 22218 823395	0.08327 67415 76704 748725
0.91223 44282 51325 905868	0.06267 20483 34109 063570
0.96397 19272 77913 791268	0.04060 14298 00386 941331
0.99312 85991 85094 924786	0.01761 40071 39152 118312
$n = 24$	
0.06405 68928 62605 626085	0.12793 81953 46752 156974
0.19111 88674 73616 309159	0.12583 74563 46828 296121
0.31504 26796 96163 374387	0.12167 04729 27803 391204
0.43379 35076 26045 138487	0.11550 56680 53725 601353
0.54542 14713 88839 535658	0.10744 42701 15965 634783
0.64809 36519 36975 569252	0.09761 86521 04113 888270
0.74012 41915 78554 364244	0.08619 01615 31953 275917
0.82000 19859 73902 921954	0.07334 64814 11080 305734
0.88641 55270 04401 034213	0.05929 85849 15436 780746
0.93827 45520 02732 758524	0.04427 74388 17419 806169
0.97472 85559 71309 498198	0.02853 13886 28933 663181
0.99518 72199 97021 360180	0.01234 12297 99987 199547

Compiled from P. Davis and P. Rabinowitz, Abscissas and weights for Gaussian quadratures of high order, J. Research NBS 56, 35-37, 1956, RP2645; P. Davis and P. Rabinowitz, Additional abscissas and weights for Gaussian quadratures of high order. Values for $n = 64$, 80, and 96, J. Research NBS 60, 613-614, 1958, RP2875; and A. N. Lowan, N. Davids, and A. Levenson, Table of the zeros of the Legendre polynomials of order 1-16 and the weight coefficients for Gauss' mechanical quadrature formula, Bull. Amer. Math. Soc. 48, 739-743, 1942 (with permission).