APPENDIX A

The Shapiro–Wilk Test for Normality

Shapiro and Wilk (1965) introduced the test for normality described below. Monte Carlo comparisons by Shapiro *et al.* (1968) indicate that it has more power than other procedures for checking the goodness-of-fit of normal distributions to data.

Suppose you have n observations $X_1, X_2, \ldots, X_n$. Proceed as follows:

(1) Sort the X_i so that $X_i > X_{i-1}$, $i = 2, 3, \ldots, n$.

(2) Calculate

$$b = \sum_{i=1}^{[n/2]} (X_{n-i+1} - X_i) a_{in},$$

where the a_{in} are given in Figure A.1.

i \ n	2	3	4	5	6	7	8	9	10	
1	0.7071	0.7071	0.6872	0.6646	0.6431	0.6233	0.6052	0.5888	0.5739	
2	—	0.0000	0.1677	0.2413	0.2806	0.3031	0.3164	0.3244	0.3291	
3	—	—	—	0.0000	0.0875	0.1401	0.1743	0.1976	0.2141	
4	—	—	—	—	—	0.0000	0.0561	0.0947	0.1224	
5	—	—	—	—	—	—	—	0.0000	0.0399	
i \ n	11	12	13	14	15	16	17	18	19	20
1	0.5601	0.5475	0.5359	0.5251	0.5150	0.5056	0.4968	0.4886	0.4808	0.4734
2	0.3315	0.3325	0.3325	0.3318	0.3306	0.3290	0.3273	0.3253	0.3232	0.3211
3	0.2260	0.2347	0.2412	0.2460	0.2495	0.2521	0.2540	0.2553	0.2561	0.2565
4	0.1429	0.1586	0.1707	0.1802	0.1878	0.1939	0.1988	0.2027	0.2059	0.2085
5	0.0695	0.0922	0.1099	0.1240	0.1353	0.1447	0.1524	0.1587	0.1641	0.1686
6	0.0000	0.0303	0.0539	0.0727	0.0880	0.1005	0.1109	0.1197	0.1271	0.1334
7	—	—	0.0000	0.0240	0.0433	0.0593	0.0725	0.0837	0.0932	0.1013
8	—	—	—	—	0.0000	0.0196	0.0359	0.0496	0.0612	0.0711
9	—	—	—	—	—	—	0.0000	0.0163	0.0303	0.0422
10	—	—	—	—	—	—	—	—	0.0000	0.0140

(*continued*)

i \ n	21	22	23	24	25	26	27	28	29	30
1	0.4643	0.4590	0.4542	0.4493	0.4450	0.4407	0.4366	0.4328	0.4291	0.4254
2	0.3185	0.3156	0.3126	0.3098	0.3069	0.3043	0.3018	0.2992	0.2968	0.2944
3	0.2578	0.2571	0.2563	0.2554	0.2543	0.2533	0.2522	0.2510	0.2499	0.2487
4	0.2119	0.2131	0.2139	0.2145	0.2148	0.2151	0.2152	0.2151	0.2150	0.2148
5	0.1736	0.1764	0.1787	0.1807	0.1822	0.1836	0.1848	0.1857	0.1864	0.1870
6	0.1399	0.1443	0.1480	0.1512	0.1539	0.1563	0.1584	0.1601	0.1616	0.1630
7	0.1092	0.1150	0.1201	0.1245	0.1283	0.1316	0.1346	0.1372	0.1395	0.1415
8	0.0804	0.0878	0.0941	0.0997	0.1046	0.1089	0.1128	0.1162	0.1192	0.1219
9	0.0530	0.0618	0.0696	0.0764	0.0823	0.0876	0.0923	0.0965	0.1002	0.1036
10	0.0263	0.0368	0.0459	0.0539	0.0610	0.0672	0.0728	0.0778	0.0822	0.0862
11	0.0000	0.0122	0.0228	0.0321	0.0403	0.0476	0.0540	0.0598	0.0650	0.0697
12	—	—	0.0000	0.0107	0.0200	0.0284	0.0358	0.0424	0.0483	0.0537
13	—	—	—	—	0.0000	0.0094	0.0178	0.0253	0.0320	0.0381
14	—	—	—	—	—	—	0.0000	0.0084	0.0159	0.0227
15	—	—	—	—	—	—	—	—	0.0000	0.0076

i \ n	31	32	33	34	35	36	37	38	39	40
1	0.4220	0.4188	0.4156	0.4127	0.4096	0.4068	0.4040	0.4015	0.3989	0.3964
2	0.2921	0.2898	0.2876	0.2854	0.2834	0.2813	0.2794	0.2774	0.2755	0.2737
3	0.2475	0.2463	0.2451	0.2439	0.2427	0.2415	0.2403	0.2391	0.2380	0.2368
4	0.2145	0.2141	0.2137	0.2132	0.2127	0.2121	0.2116	0.2110	0.2104	0.2098
5	0.1874	0.1878	0.1880	0.1882	0.1883	0.1883	0.1883	0.1881	0.1880	0.1878
6	0.1641	0.1651	0.1660	0.1667	0.1673	0.1678	0.1683	0.1686	0.1689	0.1691
7	0.1433	0.1449	0.1463	0.1475	0.1487	0.1496	0.1505	0.1513	0.1520	0.1526
8	0.1243	0.1265	0.1284	0.1301	0.1317	0.1331	0.1344	0.1356	0.1366	0.1376
9	0.1066	0.1093	0.1118	0.1140	0.1160	0.1179	0.1196	0.1211	0.1225	0.1237
10	0.0899	0.0931	0.0961	0.0988	0.1013	0.1036	0.1056	0.1075	0.1092	0.1108
11	0.0739	0.0777	0.0812	0.0844	0.0873	0.0900	0.0924	0.0947	0.0967	0.0986
12	0.0585	0.0629	0.0669	0.0706	0.0739	0.0770	0.0798	0.0824	0.0848	0.0870
13	0.0435	0.0485	0.0530	0.0572	0.0610	0.0645	0.0677	0.0706	0.0733	0.0759
14	0.0289	0.0344	0.0395	0.0441	0.0484	0.0523	0.0559	0.0592	0.0622	0.0651
15	0.0144	0.0206	0.0262	0.0314	0.0361	0.0404	0.0444	0.0481	0.0515	0.0546
16	0.0000	0.0068	0.0131	0.0187	0.0239	0.0287	0.0331	0.0372	0.0409	0.0444
17	—	—	0.0000	0.0062	0.0119	0.0172	0.0220	0.0264	0.0305	0.0343
18	—	—	—	—	0.0000	0.0057	0.0110	0.0158	0.0203	0.0244
19	—	—	—	—	—	—	0.0000	0.0053	0.0101	0.0146
20	—	—	—	—	—	—	—	—	0.0000	0.0049

i \ n	41	42	43	44	45	46	47	48	49	50
1	0.3940	0.3917	0.3894	0.3872	0.3850	0.3830	0.3808	0.3789	0.3770	0.3751
2	0.2719	0.2701	0.2684	0.2667	0.2651	0.2635	0.2620	0.2604	0.2589	0.2574
3	0.2357	0.2345	0.2334	0.2323	0.2313	0.2302	0.2291	0.2281	0.2271	0.2260
4	0.2091	0.2085	0.2078	0.2072	0.2065	0.2058	0.2052	0.2045	0.2038	0.2032
5	0.1876	0.1874	0.1871	0.1868	0.1865	0.1862	0.1859	0.1855	0.1851	0.1847
6	0.1693	0.1694	0.1695	0.1695	0.1695	0.1695	0.1695	0.1693	0.1692	0.1691
7	0.1531	0.1535	0.1539	0.1542	0.1545	0.1548	0.1550	0.1551	0.1553	0.1554
8	0.1384	0.1392	0.1398	0.1405	0.1410	0.1415	0.1420	0.1423	0.1427	0.1430
9	0.1249	0.1259	0.1269	0.1278	0.1286	0.1293	0.1300	0.1306	0.1312	0.1317
10	0.1123	0.1136	0.1149	0.1160	0.1170	0.1180	0.1189	0.1197	0.1205	0.1212
11	0.1004	0.1020	0.1035	0.1049	0.1062	0.1073	0.1085	0.1095	0.1105	0.1113
12	0.0891	0.0909	0.0927	0.0943	0.0959	0.0972	0.0986	0.0998	0.1010	0.1020
13	0.0782	0.0804	0.0824	0.0842	0.0860	0.0876	0.0892	0.0906	0.0919	0.0932
14	0.0677	0.0701	0.0724	0.0745	0.0765	0.0783	0.0801	0.0817	0.0832	0.0846
15	0.0575	0.0602	0.0628	0.0651	0.0673	0.0694	0.0713	0.0731	0.0748	0.0764
16	0.0476	0.0506	0.0534	0.0560	0.0584	0.0607	0.0628	0.0648	0.0667	0.0685
17	0.0379	0.0411	0.0442	0.0471	0.0497	0.0522	0.0546	0.0568	0.0588	0.0608
18	0.0283	0.0318	0.0352	0.0383	0.0412	0.0439	0.0465	0.0489	0.0511	0.0532
19	0.0188	0.0227	0.0263	0.0296	0.0328	0.0357	0.0385	0.0411	0.0436	0.0459
20	0.0094	0.0136	0.0175	0.0211	0.0245	0.0277	0.0307	0.0335	0.0361	0.0386
21	0.0000	0.0045	0.0087	0.0126	0.0163	0.0197	0.0229	0.0259	0.0288	0.0314
22	—	—	0.0000	0.0042	0.0081	0.0118	0.0153	0.0185	0.0215	0.0244
23	—	—	—	—	0.0000	0.0039	0.0076	0.0111	0.0143	0.0174
24	—	—	—	—	—	—	0.0000	0.0037	0.0071	0.0104
25	—	—	—	—	—	—	—	—	0.0000	0.0035

Figure A.1. Coefficients a_{in} for the Shapiro–Wilk test. (Reproduced by permission from *Biometrika*.)

(3) Calculate

$$W_n = b^2/[(n-1)s^2],$$

where s^2 is the sample variance of the X_i. Percentage points for W_n are listed in Figure A.2. For normal data W_n should be near 1; reject the assumption of normality if W_n is small.

	Level								
n	0.01	0.02	0.05	0.10	0.50	0.90	0.95	0.98	0.99
3	0.753	0.756	0.767	0.789	0.959	0.998	0.999	1.000	1.000
4	0.687	0.707	0.748	0.792	0.935	0.987	0.992	0.996	0.997
5	0.686	0.715	0.762	0.806	0.927	0.979	0.986	0.991	0.993
6	0.713	0.743	0.788	0.826	0.927	0.974	0.981	0.986	0.989
7	0.730	0.760	0.803	0.838	0.928	0.972	0.979	0.985	0.988
8	0.749	0.778	0.818	0.851	0.932	0.972	0.978	0.984	0.987
9	0.764	0.791	0.829	0.859	0.935	0.972	0.978	0.984	0.986
10	0.781	0.806	0.842	0.869	0.938	0.972	0.978	0.983	0.986
11	0.792	0.817	0.850	0.876	0.940	0.973	0.979	0.984	0.986
12	0.805	0.828	0.859	0.883	0.943	0.973	0.979	0.984	0.986
13	0.814	0.837	0.866	0.889	0.945	0.974	0.979	0.984	0.986
14	0.825	0.846	0.874	0.895	0.947	0.975	0.980	0.984	0.986
15	0.835	0.855	0.881	0.901	0.950	0.975	0.980	0.984	0.987
16	0.844	0.863	0.887	0.906	0.952	0.976	0.981	0.985	0.987
17	0.851	0.869	0.892	0.910	0.954	0.977	0.981	0.985	0.987
18	0.858	0.874	0.897	0.914	0.956	0.978	0.982	0.986	0.988
19	0.863	0.879	0.901	0.917	0.957	0.978	0.982	0.986	0.988
20	0.868	0.884	0.905	0.920	0.959	0.979	0.983	0.986	0.988
21	0.873	0.888	0.908	0.923	0.960	0.980	0.983	0.987	0.989
22	0.878	0.892	0.911	0.926	0.961	0.980	0.984	0.987	0.989
23	0.881	0.895	0.914	0.928	0.962	0.981	0.984	0.987	0.989
24	0.884	0.898	0.916	0.930	0.963	0.981	0.984	0.987	0.989
25	0.888	0.901	0.918	0.931	0.964	0.981	0.985	0.988	0.989
26	0.891	0.904	0.920	0.933	0.965	0.982	0.985	0.988	0.989
27	0.894	0.906	0.923	0.935	0.965	0.982	0.985	0.988	0.990
28	0.896	0.908	0.924	0.936	0.966	0.982	0.985	0.988	0.990
29	0.898	0.910	0.926	0.937	0.966	0.982	0.985	0.988	0.990
30	0.900	0.912	0.927	0.939	0.967	0.983	0.985	0.988	0.900
31	0.902	0.914	0.929	0.940	0.967	0.983	0.986	0.988	0.990
32	0.904	0.915	0.930	0.941	0.968	0.983	0.986	0.988	0.990
33	0.906	0.917	0.931	0.942	0.968	0.983	0.986	0.989	0.990
34	0.908	0.919	0.933	0.943	0.969	0.983	0.986	0.989	0.990
35	0.910	0.920	0.934	0.944	0.969	0.984	0.986	0.989	0.990
36	0.912	0.922	0.935	0.945	0.970	0.984	0.986	0.989	0.990
37	0.914	0.924	0.936	0.946	0.970	0.984	0.987	0.989	0.990
38	0.916	0.925	0.938	0.947	0.971	0.984	0.987	0.989	0.990
39	0.917	0.927	0.939	0.948	0.971	0.984	0.987	0.989	0.991
40	0.919	0.928	0.940	0.949	0.972	0.985	0.987	0.989	0.991
41	0.920	0.929	0.941	0.950	0.972	0.985	0.987	0.989	0.991
42	0.922	0.930	0.942	0.951	0.972	0.985	0.987	0.989	0.991
43	0.923	0.932	0.943	0.951	0.973	0.985	0.987	0.990	0.991
44	0.924	0.933	0.944	0.952	0.973	0.985	0.987	0.990	0.991
45	0.926	0.934	0.945	0.953	0.973	0.985	0.988	0.990	0.991
46	0.927	0.935	0.945	0.953	0.974	0.985	0.988	0.990	0.991
47	0.928	0.936	0.946	0.954	0.974	0.985	0.988	0.990	0.991
48	0.929	0.937	0.947	0.954	0.974	0.985	0.988	0.990	0.991
49	0.929	0.937	0.947	0.955	0.974	0.985	0.988	0.990	0.991
50	0.930	0.938	0.947	0.955	0.974	0.985	0.988	0.990	0.991

Figure A.2. Percentage points for W_n for the Shapiro–Wilk test. (Reproduced by permission from *Biometrika*.)

APPENDIX L

Routines for Random Number Generation

This appendix contains examples of generators, and of routines for calculating and inverting cdfs. Many of the routines presented are discussed in Chapters 5 and 6.

All the programs are written in ANSI Fortran. To make them as useful as possible, we have only used language features which are in both the old standard Fortran IV and the new official subset of Fortran 77 [ANSI (1978)]. All the examples have been tested on at least two machines. Error reporting is summary: in most cases a flag called IFAULT is set to a nonzero value if an error is detected, the code used depending on the routine at fault. Names follow the Fortran type conventions, and only dimensioned variables are explicitly declared. The comment AUXILIARY ROUTINES refers to routines to be found elsewhere in the appendix.

The appendix contains the following routines:

Functions

1. ALOGAM Evaluates the natural logarithm of $\Gamma(x)$ (used to set up further calls).
2. BETACH Generates a beta deviate using the method of Cheng (1978)—see Section 5.3.10.
3. BETAFX Generates a beta deviate using the method of Fox (1963)—see Section 5.3.10.
4. BOXNRM Generates a standard normal deviate using the Box–Muller method—see Section 5.2.10.
5. D2 Computes $(\exp(x) - 1)/x$ accurately (used as an auxiliary routine).

6. GAMAIN Computes the incomplete gamma ratio (used as an auxiliary routine).
7. IALIAS Generates a discrete random variate by the alias method—see Section 5.2.8. Initialization is carried out by the subroutine ALSETP.
8. IVDISC Inverts a discrete cdf using the method of Ahrens and Kohrt—see Section 5.2.1. Initialization is carried out by the subroutine VDSETP.
9. IVUNIF Generates an integer uniform over an interval $(1, n)$ using inversion—see Section 6.7.1.
10. LUKBIN Generates a discrete random variate using the method of Fox (1978b)—see Section 5.2.7. Initialization is carried out by the subroutine SETLKB.
11. OLDRND Generates uniform random numbers using the method of Schrage (1979)—see Section 6.5.2.
12. REMPIR Generates a variate from an empirical distribution with an exponential tail—see Sections 4.6 and 5.2.4.
13. RGKM3 Generates a gamma variate using rejection—see Section 5.3.9.
14. RGS Generates a gamma variate using rejection—see Section 5.3.9.
15. RSTAB Generates a stable Paretian variate using the method of Chambers *et al.* (1976)—see Section 5.3.3.
16. SUMNRM Generates a standard normal deviate by a not very quick and unnecessarily dirty method: its use is not recommended.
17. TAN Computes $\tan(x)$ (used as an auxiliary routine).
18. TAN2 Computes $\tan(x)/x$ (used as an auxiliary routine).
19. TRPNRM Generates a standard normal deviate using the method of Ahrens and Dieter—see Example 5.2.2.
20. UNIF Generates uniform random numbers. The code is fast and portable—see Section 6.5.2.
21. VBETA Calculates the inverse of the beta cdf—see Section 5.3.10.
22. VCAUCH Calculates the inverse of a standard Cauchy cdf—see Section 5.3.4.
23. VCHI2 Calculates the inverse of the chi-square cdf—see Section 5.3.11. This routine can also be used to generate Erlang and gamma variates—see Section 5.3.9.
24. VCHISQ Calculates the inverse of the chi-square cdf—see Section 5.3.11 (faster but less accurate than VCHI2). This routine can also be used to generate Erlang and gamma variates—see Section 5.3.9.
25. VF Calculates the inverse of the F cdf—see Section 5.3.12.
26. VNHOMO Generates a deviate from a nonhomogeneous Poisson distribution—see Section 5.3.18.

27. VNORM Calculates the inverse of the standard normal cdf using a rational approximation.
28. VSTUD Calculates the inverse of the t cdf—see Section 5.3.13.

Subroutines

29. BINSRH Performs a binary search in a tabulated cdf.
30. FNORM Calculates areas under a standard normal curve, and also the normal loss function.
31. TBINOM Tabulates the binomial cdf.
32. TPOISN Tabulates the cdf of a truncated Poisson.

```
      FUNCTION ALOGAM( Y)
C
C  THIS PROCEDURE EVALUATES THE NATURAL LOGARITHM OF GAMMA(Y)
C  FOR ALL Y>0, ACCURATE TO APPROXIMATELY MACHINE PRECISION
C  OR 10 DECIMAL DIGITS, WHICHEVER IS LESS.
C  STIRLING'S FORMULA IS USED FOR THE CENTRAL POLYNOMIAL PART
C  OF THE PROCEDURE.
C  BASED ON ALG. 291 IN CACM, VOL. 9, NO. 9 SEPT. 66, P. 684,
C  BY M. C. PIKE AND I. D. HILL.
C
      X = Y
      IF ( X .GE. 7.0) GO TO 20
      F = 1.0
      Z = X - 1.0
      GO TO 7
    5 X = Z
      F = F* Z
    7 Z = Z + 1.
      IF ( Z .LT. 7) GO TO 5
      X = X + 1.
      F = - ALOG( F)
      GO TO 30
   20 F = 0.
   30 Z = ( 1.0 / X) ** 2.0
      ALOGAM = F + (X - 0.5) * ALOG(X) - X + 0.918938533204673 +
     X        (((-0.000595238095238 * Z + 0.000793650793651) * Z -
     X        0.002777777777778) * Z + 0.083333333333333) / X
      RETURN
      END
```

Figure L.1. The function ALOGAM.

```
      FUNCTION BETACH( IY, A, B, CON)
      REAL CON( 3)
C
C  GENERATE A BETA DEVIATE
C  REF.:  CHENG, R., GENERATING BETA VARIATES WITH NONINTEGRAL
C  SHAPE PARAMETERS,  CACM, VOL. 21,  APRIL, 1978
C
C  INPUTS:
C    A, B = THE TWO PARAMETERS OF THE STANDARD BETA,
C    CON( ) = A REAL WORK VECTOR OF LENGTH AT LEAST 3.
C    AT THE FIRST CALL CON( 1) SHOULD BE 0.
C    IT IS SET TO A FUNCTION OF A & B FOR USE IN SUBSEQUENT CALLS
C    IY = A RANDOM NUMBER SEED.
C
C  AUXILIARY ROUTINE:
C    UNIF
C
      IF ( CON( 1) .GT. 0.) GO TO 200
      CON( 1) = AMIN1( A, B)
      IF ( CON( 1) .GT. 1.) GO TO 100
      CON( 1) = 1./ CON( 1)
      GO TO 150
  100 CON( 1) = SQRT(( A + B - 2.)/( 2.* A* B- A- B))
  150 CON( 2)= A + B
      CON ( 3) = A + 1./ CON( 1)
C
C  SUBSEQUENT ENTRIES ARE HERE
C
  200 U1 = UNIF(IY)
      U2 = UNIF( IY)
      V = CON( 1)* ALOG( U1/(1.- U1))
      W = A* EXP( V)
      IF (( CON( 2))*ALOG(( CON( 2))/(B+W)) + (CON( 3))
     X *V - 1.3862944 .LT. ALOG( U1*U1*U2)) GO TO 200
      BETACH = W/( B + W)
      RETURN
      END
```

Figure L.2. The function BETACH.

```
      FUNCTION BETAFX( IX, NA, NB, WORK)
      REAL WORK( NA)
C
C  GENERATE A BETA DEVIATE
C  REF.: FOX, B.(1963), GENERATION OF RANDOM SAMPLES FROM THE
C  BETA AND F DISTRIBUTIONS, TECHNOMETRICS, VOL. 5, 269-270.
C
C  INPUTS:
C    IX = A RANDOM NUMBER SEED,
C    NA, NB = PARAMETERS OF THE BETA DISTRIBUTION,
C    MUST BE INTEGER
C    WORK( ) = A WORK VECTOR OF LENGTH AT LEAST NA + NB.
C
C  OUTPUTS:
C    BETAFX = A DEVIATE FROM THE RELEVANT DISTRIBUTION
C
C  AUXILIARY ROUTINE:
C    UNIF
C
C  SETUP WORK VECTOR FOR SORTING
C
      DO 100 I =  1, NA
      WORK( I) = 2.
  100 CONTINUE
      NAB = NA + NB - 1
C
C  GENERATE NAB UNIFORMS AND FIND THE NA-TH SMALLEST
C
      DO 200 I = 1, NAB
      FY = UNIF( IX)
      IF ( FY .GE. WORK( 1)) GO TO 200
      IF ( NA .LE. 1) GO TO 170
      DO 160 K = 2, NA
      IF ( FY .LT.  WORK( K)) GO TO 150
      WORK( K - 1) = FY
      GO TO 200
  150 WORK( K - 1) = WORK( K)
  160 CONTINUE
  170 WORK( NA) = FY
  200 CONTINUE
C
      BETAFX = WORK( 1)
      RETURN
      END
```

Figure L.3. The function BETAFX.

```
      FUNCTION BOXNRM( IY, U1)
C
C  RETURN A UNIT NORMAL(OR GAUSSIAN) RANDOM VARIABLE
C  USING THE BOX-MULLER METHOD.
C
C  INPUT:
C  IY = RANDOM NUMBER SEED
C  U1 = A WORK SCALAR WHICH SHOULD BE > 254 ON FIRST CALL.
C     IT STORES DATA FOR SUBSEQUENT CALLS
C
C  AUXILIARY ROUTINE:
C    UNIF
C
      IF ( U1 .GE. 254.) GO TO 100
C
C  A DEVIATE IS LEFT FROM LAST TIME
C
      BOXNRM =  U1
C
C  INDICATE THAT NONE LEFT. NOTE THAT PROBABILITY THAT
C  A DEVIATE >  254 IS EXTREMELY SMALL.
C
      U1 = 255.
      RETURN
C
C  TIME TO GENERATE A PAIR
C
  100 U1 = UNIF( IY)
      U1 = SQRT(-2.* ALOG( U1))
      U2 = 6.2831852* UNIF( IY)
C
C  USE ONE...
C
      BOXNRM = U1* COS( U2)
C
C  SAVE THE OTHER FOR THE NEXT PASS
C
      U1 = U1* SIN( U2)
      RETURN
      END
```

Figure L.4. The function BOXNRM.

```
      FUNCTION D2( Z)
C
C  EVALUATE ( EXP( X ) - 1)/X
C
      DOUBLE PRECISION P1, P2, Q1, Q2, Q3, PV, ZZ
C
C  ON COMPILERS WITHOUT DOUBLE PRECISION THE ABOVE MUST BE
C  REPLACED BY REAL P1, ETC., AND THE DATA INITIALIZATION
C  MUST USE E RATHER THAN D FORMAT.
C
      DATA P1, P2, Q1, Q2, Q3/.84006685253648323 9D3,
     X .20001114158996456 9D2,
     X .16801337050792664 8D4,
     X .18013370407390023D3,
     X 1.DO/
C
C  THE APPROXIMATION 1801 FROM HART ET. AL.(1968, P. 213)
C
      IF ( ABS( Z) .GT. 0.1) GO TO 100
      ZZ = Z * Z
      PV = P1 + ZZ* P2
      D2 = 2.DO* PV/( Q1+ ZZ*( Q2+ ZZ* Q3) - Z* PV)
      RETURN
  100 D2 = ( EXP( Z) - 1.) / Z
      RETURN
      END
```

Figure L.5. The function D2.

```
      FUNCTION GAMAIN( X, P, G, IFAULT)
C
C  ALGORITHM AS 32 J.R.STATIST.SOC. C. (1970) VOL.19 NO.3
C
C  COMPUTES INCOMPLETE GAMMA RATIO FOR POSITIVE VALUES OF
C  ARGUMENTS X AND P.  G MUST BE SUPPLIED AND SHOULD BE EQUAL TO
C  LN(GAMMA(P)).
C
C  IFAULT = 1 IF P.LE.0 ELSE 2 IF X.LT.0 ELSE 0.
C  USES SERIES EXPANSION IF P.GT.X OR X.LE.1, OTHERWISE A
C  CONTINUED FRACTION APPROXIMATION.
C
      DIMENSION PN(6)
C
C  DEFINE ACCURACY AND INITIALIZE
C
      DATA ACU/1.E-8/, OFLO/1.E30/
      GIN=0.0
      IFAULT=0
C
C  TEST FOR ADMISSIBILITY OF ARGUMENTS
C
      IF(P.LE.0.0) IFAULT=1
      IF(X.LT.0.0) IFAULT=2
      IF(IFAULT.GT.0.OR.X.EQ.0.0) GO TO 50
      FACTOR=EXP(P*ALOG(X)-X-G)
      IF(X.GT.1.0.AND.X.GE.P) GO TO 30
```

(continued)

```
C
C  CALCULATION BY SERIES EXPANSION
C
      GIN=1.0
      TERM=1.0
      RN=P
   20 RN=RN+1.0
      TERM=TERM*X/RN
      GIN=GIN+TERM
      IF(TERM.GT.ACU) GO TO 20
      GIN=GIN*FACTOR/P
      GO TO 50
C
C  CALCULATION BY CONTINUED FRACTION
C
   30 A=1.0-P
      B=A+X+1.0
      TERM=0.0
      PN(1)=1.0
      PN(2)=X
      PN(3)=X+1.0
      PN(4)=X*B
      GIN=PN(3)/PN(4)
   32 A=A+1.0
      B=B+2.0
      TERM=TERM+1.0
      AN=A*TERM
      DO 33 I=1,2
   33 PN(I+4)=B*PN(I+2)-AN*PN(I)
      IF(PN(6).EQ.0.0) GO TO 35
      RN=PN(5)/PN(6)
      DIF=ABS(GIN-RN)
      IF(DIF.GT.ACU) GO TO 34
      IF(DIF.LE.ACU*RN) GO TO 42
   34 GIN=RN
   35 DO 36 I=1,4
   36 PN(I)=PN(I+2)
      IF(ABS(PN(5)).LT.OFLO) GO TO 32
      DO 41 I=1,4
   41 PN(I)=PN(I)/OFLO
      GO TO 32
   42 GIN=1.0-FACTOR*GIN
C
   50 GAMAIN=GIN
      RETURN
      END
```

Figure L.6. The function GAMAIN.

```
      FUNCTION IALIAS( PHI, N, A, R)
      INTEGER A( N)
      REAL R( N)
C
C  GENERATE A DISCRETE R.V. BY THE ALIAS METHOD
C
C  INPUTS:
C    PHI = NUMBER BETWEEN 0 AND 1
C    N = RANGE OF R. V., I.E., 1, 2, ..., N
C    A( I) = ALIAS OF I
C    R( I) = ALIASING PROBABILITY
C
C  REF.: KRONMAL AND PETERSON, THE AMERICAN STATISTICIAN,
C     VOL 33, 1979.
C
      V = PHI *N
      JALIAS = V + 1.
      IF( JALIAS - V .GT. R( JALIAS)) JALIAS = A( JALIAS)
      IALIAS = JALIAS
      RETURN
      END



      SUBROUTINE ALSETP( N, P, A, R, L)
      INTEGER A( N), L( N)
      REAL P( N), R( N)
C
C  SETUP TABLES FOR USING THE ALIAS METHOD OF GENERATING
C  DISCRETE RANDOM VARIABLES USING FUNCTION IALIAS.
C
C  INPUTS:  N = RANGE OF THE RANDOM VARIABLE, I.E. 1, 2,..., N
C    P( I) = PROB( R.V. = I)
C    L(  ) A WORK VECTOR OF SIZE N
C
C  OUTPUTS:  A( I) = ALIAS OF I
C    R( I) ALIASING PROBABILITY OF I
C
      LOW = 0
      MID = N+ 1
C
C  PUT AT THE BOTTOM, OUTCOMES FOR WHICH THE UNIFORM DISTRIBUTION
C  ASSIGNS TOO MUCH PROBABILITY. PUT THE REST AT THE TOP.
C
      DO 100 I = 1, N
      A( I) = I
      R( I) = N*P( I)
      IF ( R( I) .GE. 1.) GO TO 50
C
C  TOO MUCH PROBABILITY ASSIGNED TO I
C
      LOW = LOW + 1
      L(LOW) = I
      GO TO 100
C
C  TOO LITTLE PROBABILITY ASSIGNED TO I
C
   50 MID = MID - 1
      L( MID) = I
  100 CONTINUE
```

(*continued*)

```
C
C  NOW GIVE THE OUTCOMES AT THE BOTTOM(WITH TOO MUCH PROBABILITY)
C  AN ALIAS FROM THE TOP.
C
      N1 = N - 1
      DO 200 I = 1, N1
       K = L( MID)
      J = L( I)
      A( J)  = K
      R( K) = R( K) + R( J) - 1.
      IF ( R( K) .GE. 1.) GO TO 200
      MID = MID + 1
  200 CONTINUE
      RETURN
      END
```

Figure L.7. The function IALIAS and its initialization subroutine ALSETP.

```
      FUNCTION IVDISC( U, N, M, CDF, LUCKY)
      REAL CDF( N)
      INTEGER LUCKY( M)
C
C  INVERT A DISCRETE CDF USING AN INDEX VECTOR TO SPEED THINGS UP
C  REF. AHRENS AND KOHRT,  COMPUTING, VOL 26, P19, 1981.
C
C  INPUTS:
C    U = A NUMBER 0 <= U <= 1.
C    N = SIZE OF CDF; = RANGE OF THE RANDOM VARIABLE; = 1,2 ...,N
C    M = SIZE OF LUCKY
C    CDF = THE CDF VECTOR.
C    LUCKY = THE POINTER VECTOR
C
C  OUTPUT:
C    IVDISC = F INVERSE OF U
C
C  TAKE CARE OF CASE U = 1.
C
      IF ( U .LT. CDF( N)) GO TO 50
      IVDISC = N
      RETURN
C
   50 NDX = U* M + 1.
      JVDISC = LUCKY( NDX)
C
C  IS U IN A GRID INTERVAL FOR WHICH THERE IS A JVDISC
C  SUCH THAT CDF( JVDISC - 1) < INTERVAL <= CDF( JVDISC) ?
C
      IF ( JVDISC .GT. 0) GO TO 300
C
C  HARD LUCK, WE MUST SEARCH THE CDF FOR SMALLEST JVDISC
C  SUCH THAT CDF( JVDISC) >= U.
C
      JVDISC = - JVDISC
      GO TO 200
  100 JVDISC = JVDISC + 1
  200 IF ( CDF( JVDISC) .LT. U) GO TO 100
  300 IVDISC = JVDISC
      RETURN
      END



      SUBROUTINE VDSETP( N, M, CDF, LUCKY)
      REAL CDF( N)
      INTEGER LUCKY( M)
C
C  SETUP POINTER VECTOR LUCKY FOR INDEXED INVERSION
C  OF DISCRETE CDF BY THE FUNCTION IVDISC.
C
C  INPUTS:
C    N = NO. POINTS IN CDF
C    M = NO. LOCATIONS ALLOCATED FOR POINTERS
C    CDF = THE CDF VECTOR
C
C  OUTPUTS:
C    LUCKY = THE POINTER VECTOR; > 0 IMPLIES THIS POINTER
C    GRID INTERVAL FALLS COMPLETELY WITHIN SOME CDF INTERVAL;
C    < 0 IMPLIES THE GRID INTERVAL IS SPLIT BY A CDF POINT.
C
C  TAKE CARE OF DEGENERATE CASE OF N = 1
```

(*continued*)

```
C
      DO 25 I = 1, M
      LUCKY( I)= 1
   25 CONTINUE
      IF ( N .EQ. 1) RETURN
C
C  INITIALIZE FOR GENERAL CASE
C
      NOLD = M
      I = N
      DO 300 II = 2, N
      I = I - 1
C
C  FIND THE GRID INTERVAL IN WHICH CDF( I) FALLS
C
      NDX = CDF( I)*M +1
      LUCKY( NDX) = - I
C
C  ANY GRID INTERVALS FALLING STRICTLY BETWEEN THIS
C  CDF POINT AND THE THE NEXT HIGHER, MAP TO THE NEXT HIGHER
C
      ND1 = NDX +1
      IF ( ND1 .GT. NOLD) GO TO 200
      DO 100 K = ND1, NOLD
      LUCKY( K) = I + 1
  100 CONTINUE
  200 NOLD = NDX - 1
  300 CONTINUE
      RETURN
      END
```

Figure L.8. The function IVDISC and its initialization subroutine VDSETP.

```
      FUNCTION IVUNIF( IX, N)
C
C  PORTABLE CODE TO GENERATE AN INTEGER UNIFORM OVER
C  THE INTERVAL 1,N USING INVERSION
C
C  INPUTS:
C    IX = RANDOM NUMBER SEED
C    M = LARGEST VALUE FOR IX + 1
C
C  AUXILIARY ROUTINE:
C    UNIF
C
C  THE RANDOM NUMBER GENERATOR IS ASSUMED TO USE THIS MODULUS
      DATA M/2147483647/
C
      JUNK = UNIF( IX)
C
C  WE WANT INTEGER PART OF IX/(M/N) USING INFINITE PRECISION.
C  FORTRAN TRUNCATION UNDERSTATES M / N, THUS, FORTRAN
C  MAY OVERSTATE WHEN IT COMPUTES IX/( M/ N).  SO DECREASE
C  JTRY UNTIL JTRY <= IX/(M/N), I.E. M/IX <= N/JTRY.
C
      JTRY = IX/( M/ N)
      IF ( JTRY .LE. 0) GO TO 500
      GO TO 200
  100 JTRY = JTRY - 1
C
C  NOW CHECK IF M/IX <= N/JTRY
C
  200 I = M
      J = IX
      K = N
      L = JTRY
C
C  MORE GENERALLY, IS I/J <= K/L, WHERE I>=J, K>= L
C  IF THE INTEGER PARTS DIFFER THE QUESTION IS ANSWERED.
C
  300 IJI = I/ J
      KLI = K/ L
      IF ( IJI .LT. KLI) GO TO 500
      IF ( IJI .GT. KLI) GO TO 100
C
C  MUST LOOK AT THE REMAINDERS;  IS IJR/J <= KLR/L
C
      IJR = I - IJI * J
      IF ( IJR .LE. 0) GO TO 500
      KLR = K - KLI* L
      IF ( KLR .LE. 0) GO TO 100
C
C  STILL NOT RESOLVED; REPHRASE IN STANDARD FORM,
C  I.E. IS L/KLR <= J/IJR ?
C
      I = L
      K = J
      J = KLR
      L = IJR
      GO TO 300
```

(*continued*)

```
C
C  NOW ADD 1 TO PUT IN THE RANGE 1,N
C
  500 IVUNIF = JTRY + 1
      RETURN
      END
```

Figure L.9. The function IVUNIF.

```
      FUNCTION LUKBIN( PHI, N, R, T, M, NK)
      REAL R( N)
      INTEGER NK( M)
C
C  THE BUCKET-BINARY SEARCH METHOD OF FOX
C  REF. OPERATIONS RESEARCH, VOL 26, 1978
C  FOR GENERATING(QUICKLY) DISCRETE RANDOM VARIABLES
C
C  INPUTS:
C    PHI = A PROBABILITY, 0 < PHI < 1
C    N =  NO. POSSIBLE OUTCOMES FOR THE R. V.
C    R = A VECTOR SET BY SETLKB
C    T = A SCALER THRESHOLD SET BY SETLKB
C    M = NUMBER OF CELLS ALLOWED FOR THE POINTER VECTOR NK( ).
C    NK = A POINTER VECTOR SET BY SETLKB
C
C  OUTPUTS:
C    LUKBIN = RANDOM VARIABLE VALUE
C
      IF ( PHI .GE. T) GO TO 50
C
C  HERE IS THE QUICK BUCKET METHOD
C
      I = M * PHI
      LUKBIN = NK( I + 1)
      RETURN
C
C  WE MUST RESORT TO BINARY SEARCH
C
   50 IF (PHI .LE. R( 1)) GO TO 500
      MAXI = N
      MINI = 2
  100 K = ( MAXI + MINI)/ 2
      IF ( PHI .GT. R(K)) GO TO 300
      IF ( PHI .GT. R( K - 1)) GO TO 500
      MAXI = K - 1
      GO TO 100
  300 MINI = K + 1
      GO TO 100
  500 LUKBIN = K
      RETURN
      END



      SUBROUTINE SETLKB( M, N, T, CDF, NK, R)
      REAL CDF( N), R( N)
      INTEGER NK( M)
C
C  SETUP FOR THE LUKBIN METHOD.  SEE FUNCTION LUKBIN
C
C  INPUTS:
C    N =  NO. POSSIBLE OUTCOMES FOR THE R. V.
C    CDF( ) = CDF OF THE R. V.
C    M = NUMBER OF CELLS ALLOWED FOR THE POINTER VECTOR NK( ).
C    M > 0.  LARGER M LEADS TO GREATER SPEED.
C
C  OUTPUTS:  T = A THRESHOLD TO BE USED BY LUKBIN
C    NK( ) = A POINTER VECTOR WHICH SAVES TIME
C    R() = A THRESHOLD PROBABILITY VECTOR.
```

(*continued*)

```
C
      EXCESS = 0.
      PO = 0.
      NLIM = 0
      NUSED = 0
      DO 400 I = 1, N
      PI = CDF( I) - PO
      LOT = M* PI
      PO = CDF( I)
      IF ( LOT .LE. 0) GO TO 200
      NUSED = NLIM + 1
      NLIM = NLIM + LOT
      DO 100 J = NUSED, NLIM
      NK( J) = I
  100 CONTINUE
  200 EXCESS = EXCESS + PI - FLOAT(LOT)/FLOAT( M)
      R( I) = EXCESS
  400 CONTINUE
      T  = FLOAT( NLIM)/ FLOAT( M)
      DO 500 I = 1, N
      R( I) = R( I) + T
  500 CONTINUE
      RETURN
      END
```

Figure L.10. The function LUKBIN and its initialization subroutine SETLKB.

```
      FUNCTION OLDRND( IY)
C
C  -2BYTE      INTEGER*4 IY
C  THE ABOVE DECLARATION MAY BE NEEDED ON SOME COMPILERS,
C  E.G., HP3000.  THE CRUCIAL CONSIDERATION IS THAT IY HAVE
C  AT LEAST 32 BIT ACCURACY.
C
C  PORTABLE RANDOM NUMBER GENERATOR: SEE SCHRAGE (1979)
C  USING THE RECURSION
C    IY = IY*A MOD P
C
C  INPUTS:
C    IY = INTEGER GREATER THAN 0 AND LESS THAN 2147483647
C
C  OUTPUTS:
C    IY = NEW PSEUDORANDOM VALUE,
C    OLDRND = A UNIFORM FRACTION BETWEEN 0 AND 1.
C
      INTEGER A,P,B15,B16,XHI,XALO,LEFTLO,FHI,K
C  -2BYTE      INTEGER*4 A,P,B15,B16,XHI,XALO,LEFTLO,FHI,K
C  THE ABOVE DECLARATION MAY BE NEEDED ON SOME COMPILERS,
C  E.G., HP3000.  THE CRUCIAL CONSIDERATION IS THAT
C  THE VARIABLES HAVE AT LEAST 32 BIT ACCURACY.
C
C  7**5, 2**15, 2**16, 2**31-1
C
      DATA A/16807/,B15/32768/,B16/65536/,P/2147483647/
C
C  CORRECT ANY BAD INPUT
C
      IF ( IY .GT. 0) GO TO 200
C
C  SET IT TO SOME VALID VALUE
C
      IY = 63887
C
C  GET 15 HI ORDER BITS OF IY
C
  200 XHI = IY/B16
C
C  GET 16 LO BITS OF IY AND FORM LO PRODUCT
C
      XALO=(IY-XHI*B16)*A
C
C  GET 15 HI ORDER BITS OF LO PRODUCT
C
      LEFTLO = XALO/B16
C
C  FORM THE 31 HIGHEST BITS OF FULL PRODUCT
C
      FHI = XHI*A + LEFTLO
C
C  GET OVERFLO PAST 31ST BIT OF FULL PRODUCT
C
      K = FHI/B15
C
C  ASSEMBLE ALL THE PARTS AND PRESUBTRACT P
C  THE PARENTHESES ARE ESSENTIAL
```

(*continued*)

```
C
      IY = (((XALO-LEFTLO*B16) - P) + (FHI-K*B15)*B16) + K
C
C  ADD P BACK IN IF NECESSARY
C
      IF (IY .LT. 0) IY = IY + P
C
C  MULTIPLY BY 1/(2**31-1)
C
      OLDRND = IY*4.656612875E-10
      RETURN
      END
```

Figure L.11. The function OLDRND.

```
      FUNCTION REMPIR( U, N, K, X, THETA, IFAULT)
      DIMENSION X(N)
C
C  INVERSION OF AN EMPIRICAL DISTRIBUTION WITH AN
C  EXPONENTIAL APPROXIMATION TO THE K LARGEST POINTS.
C
C  INPUTS:
C    U = PROBABILITY BEING INVERTED
C    N = NO. OF OBSERVATIONS
C    K = NO. OBS. TO BE USED TO GET EXPONENTIAL RIGHT TAIL
C    X = ORDERED VECTOR OF OBSERVATIONS
C    THETA = MEAN OF THE EXPONENTIAL TAIL,  < 0 ON FIRST CALL
C
C  OUTPUT:
C    REMPIR = EMPIRICAL F INVERSE OF U
C    THETA = MEAN OF THE EXPONENTIAL TAIL
C    IFAULT = 0 IF NO ERROR IN INPUT DATA, 3 OTHERWISE
C
      IF ( THETA .GT. 0.) GO TO 100
C
C  FIRST CALL; CHECK FOR ERRORS...
C
      IF (( N .LE. 0) .OR. ( K .GT. N) .OR. ( K .LT. 0)) GO TO 9000
      XO = 0.
C
C  E.G.,  IF ORDERED
C
      DO 20 I = 1, N
      IF ( X(I) .LT. XO) GO TO 9000
      XO = X(I)
   20 CONTINUE
C
C  INPUT IS OK
C
      IFAULT = 0
      IF ( K .GT. 0) GO TO 45
C
C  IT IS A PURE EMPIRICAL DISTRIBUTION, FLAG THETA
C
      THETA = .000001
      GO TO 100
C
C  CALCULATE MEAN OF EXPONENTIAL TAIL
C  TAKE CARE OF SPECIAL CASE OF PURE EXPONENTIAL
C
   45 NK = N - K + 1
C
C  IN PURE EXPONENTIAL CASE, THETA = 0.
C
      THETA = 0.
      IF ( K .LT. N) THETA = - X( N- K)*( K - .5)
      DO 50 I = NK, N
      THETA = THETA + X(I)
   50 CONTINUE
      THETA = THETA/ K
C
C  HERE IS THE CODE FOR SUBSEQUENT CALLS
C
  100 IF ( U .LE. 1. - FLOAT(K)/FLOAT(N)) GO TO 200
C
C  PULL IT BY(FROM) THE TAIL
```

(*continued*)

```
C
      XNK = 0.
      IF ( K .LT. N) XNK = X(N - K)
      REMPIR = XNK - THETA*ALOG(N*(1.- U)/ K)
      RETURN
C
C  PULL IT FROM THE EMPIRICAL PART
C
  200 V = N* U
      I = V
      XO = 0.
      IF ( I .GT. 0) XO = X(I)
      REMPIR = XO + ( V - I)*( X(I+1) - XO)
      RETURN
C
 9000 IFAULT = 3
      RETURN
      END
```

Figure L.12. The function REMPIR.

```
      FUNCTION RGKM3( ALP, WORK, K, IX )
      REAL WORK(6)
C
C  GENERATE A GAMMA VARIATE WITH PARAMETER ALP
C
C  INPUTS:
C    ALP = DISTRIBUTION PARAMETER, ALP .GT. 1
C    WORK() = VECTOR OF WORK CELLS; ON FIRST CALL WORK(1) = -1.
C    WORK() MUST BE PRESERVED BY CALLER BETWEEN CALLS.
C    K = WORK CELL. K MUST BE PRESERVED BY CALLER BETWEEN CALLS.
C    IX = RANDOM NUMBER SEED
C
C  OUTPUTS:
C    RGKM3 = A GAMMA VARIATE
C
C  REFERENCES: CHENG, R.C. AND G.M. FEAST(1979), SOME SIMPLE GAMMA
C  VARIATE GENERATORS, APPLIED STAT. VOL 28, PP. 290-295.
C  TADIKAMALLA, P.R. AND M.E. JOHNSON(1981), A COMPLETE GUIDE TO GAMMA
C  VARIATE GENERATION, AMER. J. OF MATH. AND MAN. SCI., VOL. 1, PP. 213-236.
C
      IF ( WORK(1) .EQ. ALP) GO TO 100
C
C  INITIALIZATION IF THIS IS FIRST CALL
C
      WORK(1) = ALP
      K = 1
      IF ( ALP .GT. 2.5) K = 2
      WORK(2) = ALP - 1.
      WORK(3) = ( ALP - 1./( 6.* ALP )) / WORK(2)
      WORK(4) = 2. / WORK(2)
      WORK(5) = WORK(4) + 2.
      GO TO ( 1, 11) , K
C
C  CODE FOR SUBSEQUENT CALLS STARTS HERE
C
    1 U1 = UNIF( IX)
      U2 = UNIF( IX)
      GO TO 20
   11 WORK(6) = SQRT( ALP )
   15 U1 = UNIF( IX)
      U = UNIF( IX)
      U2 = U1 + ( 1. - 1.86* U ) / WORK(6)
      IF (( U2 .LE. 0.) .OR. ( U2 .GE. 1. ) ) GO TO 15
   20 W = WORK(3) * U1 / U2
      IF (( WORK(4)* U2 - WORK(5)+ W + 1. /  W ) .LE. 0.) GO TO 200
      IF (( WORK(4)* ALOG( U2)- ALOG( W)+ W - 1.) .LT. 0.) GO TO 200
  100 GO TO ( 1, 15) , K
  200 RGKM3 = WORK(2) * W
      RETURN
      END
```

Figure L.13. The function RGKM3.

```
      FUNCTION RGS( ALP, IX)
C
C  GENERATE A GAMMA VARIATE WITH PARAMETER ALP
C
C  INPUTS:
C    ALP = DISTRIBUTION PARAMETER; ALP .LE. 1.
C    IX = RANDOM NUMBER SEED
C
C  OUTPUTS:
C    RGS = A GAMMA VARIATE
C
C  REFERENCES: AHRENS, J.H.  AND U. DIETER(1972), COMPUTER METHODS FOR
C  SAMPLING FROM GAMMA, BETA, POISSON AND BINOMIAL DISTRIBUTIONS,
C  COMPUTING, VOL. 12, PP. 223-246
C  TADIKAMALLA, P.R. AND M.E. JOHNSON(1981), A COMPLETE GUIDE TO
C  GAMMA VARIATE GENERATION, AMER. J. OF MATH. AND MAN. SCI.,
C  VOL. 1, PP. 213-236.
C
    1 U1 = UNIF( IX)
      B = ( 2.718281828 + ALP) / 2.718281828
      P = B * U1
      IF ( P .GT. 1.) GO TO 3
      X = EXP( ALOG( P) / ALP)
      U2 = UNIF( IX)
      IF ( U2 .GT. EXP( - X)) GO TO 1
      RGS = X
      RETURN
    3 X = - ALOG(( B - P) / ALP)
      U3 = UNIF( IX)
      IF ( ALOG( U3) .GT. ( ALP - 1.)* ALOG( X)) GO TO 1
      RGS = X
      RETURN
      END
```

Figure L.14. The function RGS.

```
      FUNCTION RSTAB( ALPHA, BPRIME, U, W)
C
C  GENERATE A STABLE PARETIAN VARIATE
C
C  INPUTS:
C    ALPHA = CHARACTERISTIC EXPONENT, 0<= ALPHA <= 2.
C    BPRIME = SKEWNESS IN REVISED PARAMETERIZATION, SEE REF.
C           = 0 GIVES A SYMMETRIC DISTRIBUTION.
C    U = UNIFORM VARIATE ON (0, 1.)
C    W = EXPONENTAIL DISTRIBUTED VARIATE WITH MEAN 1.
C
C  REFERENCE: CHAMBERS, J. M., C. L. MALLOWS AND B. W. STUCK(1976),
C  "A METHOD FOR SIMULATING STABLE RANDOM VARIABLES", JASA, VOL. 71,
C  NO. 354, PP. 340-344
C
C  AUXILIARY ROUTINES:
C    D2, TAN, TAN2
C
      DOUBLE PRECISION DA, DB
C
C  ON COMPILERS WITHOUT DOUBLE PRECISION THE ABOVE MUST BE
C  REPLACED BY REAL DA, DB
C
      DATA PIBY2/1.57079633E0/
      DATA THR1/0.99/
      EPS = 1. - ALPHA
C
C COMPUTE SOME TANGENTS
C
      PHIBY2 = PIBY2*( U - .5)
      A = PHIBY2* TAN2( PHIBY2)
      BB = TAN2( EPS* PHIBY2)
      B = EPS* PHIBY2* BB
      IF ( EPS .GT. -.99) TAU = BPRIME/( TAN2( EPS* PIBY2)* PIBY2)
      IF ( EPS .LE. -.99) TAU = BPRIME*PIBY2* EPS*(1. - EPS) *
     X TAN2(( 1. - EPS)* PIBY2)
C
C  COMPUTE SOME NECESSARY SUBEXPRESSIONS
C  IF PHI NEAR PI BY 2, USE DOUBLE PRECISION.
C
      IF ( A .GT. THR1) GO TO 50
C
C  SINGLE PRECISION
C
      A2 = A**2
      A2P = 1. + A2
      A2 = 1. - A2
      B2 = B**2
      B2P = 1. + B2
      B2 = 1. - B2
      GO TO 100
C
C  DOUBLE PRECISION
C
   50 DA = A
      DA = DA**2
      DB = B
      DB = DB**2
      A2 = 1.D0 - DA
      A2P = 1.D0 + DA
      B2 = 1.D0 - DB
      B2P = 1.D0 + DB
```

(*continued*)

```
C
C  COMPUTE COEFFICIENT
C
  100 Z = A2P*( B2 + 2.* PHIBY2* BB * TAU)/( W* A2 * B2P)
C
C  COMPUTE THE EXPONENTIAL-TYPE EXPRESSION
C
      ALOGZ = ALOG( Z)
      D = D2( EPS* ALOGZ/(1. - EPS))*( ALOGZ/(1.- EPS))
C
C  COMPUTE STABLE
C
      RSTAB = ( 1. + EPS* D) * 2.*(( A - B)*(1. + A* B) - PHIBY2*
     X TAU* BB*( B* A2 - 2. * A))/( A2 * B2P) + TAU* D
      RETURN
      END
```

Figure L.15. The function RSTAB.

```
      FUNCTION SUMNRM( IX)
C
C  SUM 12 UNIFORMS AND SUBTRACT 6 TO APPROXIMATE A NORMAL
C  ***NOT RECOMMENDED***
C
C  INPUT:
C    IX = A RANDOM NUMBER SEED
C
C  AUXILIARY ROUTINE:
C    UNIF
C
      SUM = - 6.
      DO 100 I = 1, 12
      SUM = SUM + UNIF( IX)
  100 CONTINUE
      SUMNRM = SUM
      RETURN
      END
```

Figure L.16. The function SUMNRM.

```
      FUNCTION TAN( XARG)
C
C  EVALUATE THE TANGENT OF XARG
C
      LOGICAL NEG, INV
      DATA P0, P1, P2, Q0, Q1, Q2
     X /.129221035E+3,-.887662377E+1,.528644456E-1,
     X .164529332E+3,-.451320561E+2,1./
C
C  THE APPROXIMATION 4283 FROM HART ET. AL.( 1968, P. 251)
C
      DATA PIBY4/.785398163E0/,PIBY2/1.57079633E0/
      DATA PI /3.14159265E0/
      NEG = .FALSE.
      INV = .FALSE.
      X = XARG
      NEG = X .LT. 0.
      X = ABS( X)
C
C  PERFORM RANGE REDUCTION IF NECESSARY
C
      IF ( X .LE. PIBY4) GO TO 50
      X = AMOD( X, PI)
      IF ( X .LE. PIBY2) GO TO 30
      NEG = .NOT. NEG
      X = PI - X
   30 IF ( X .LE. PIBY4) GO TO 50
      INV = .TRUE.
      X = PIBY2 - X
   50 X = X/ PIBY4
C
C  CONVERT TO RANGE OF RATIONAL
C
      XX = X * X
      TANT = X*( P0+ XX*( P1+ XX* P2))/( Q0 + XX*( Q1+ XX* Q2))
      IF ( NEG) TANT = - TANT
      IF ( INV) TANT = 1./ TANT
      TAN = TANT
      RETURN
      END
```

Figure L.17. The function TAN.

```
      FUNCTION TAN2( XARG)
C
C  COMPUTE TAN( XARG)/ XARG
C  DEFINED ONLY FOR ABS( XARG) .LE. PI BY 4
C  FOR OTHER ARGUMENTS RETURNS TAN( X)/X COMPUTED DIRECTLY
C
      DATA P0, P1, P2, Q0, Q1, Q2
     X /.129221035E+3,-.887662377E+1,.528644456E-1,
     X  .164529332E+3,-.451320561E+2,1.0/
C
C  THE APPROXIMATION 4283 FROM HART ET. AL(1968, P. 251)
C
      DATA PIBY4/.785398163E0/
C
      X = ABS( XARG)
      IF ( X .GT. PIBY4) GO TO 200
      X = X/ PIBY4
C
C  CONVERT TO RANGE OF RATIONAL
C
      XX = X* X
      TAN2 =( P0+ XX*( P1 + XX* P2))/( PIBY4*( Q0+ XX*( Q1+ XX* Q2)))
      RETURN
  200 TAN2 = TAN( XARG)/ XARG
      RETURN
      END
```

Figure L.18. The function TAN2.

```
      FUNCTION TRPNRM( IX)
C
C  GENERATE UNIT NORMAL DEVIATE BY COMPOSITION METHOD
C  OF AHRENS AND DIETER
C  THE  AREA UNDER THE NORMAL CURVE IS DIVIDED INTO 5
C  DIFFERENT AREAS.
C
C  INPUT:
C    IX = A RANDOM NUMBER SEED
C
C  AUXILIARY ROUTINE:
C    UNIF
C
      U = UNIF( IX)
      UO = UNIF( IX)
      IF (U .GE. .919544) GO TO 160
C
C  AREA A,  THE TRAPEZOID IN THE MIDDLE
C
      TRPNRM = 2.40376*( UO+ U*.825339)-2.11403
      RETURN
  160 IF ( U .LT. .965487) GO TO 210
C
C  AREA B
C
  180 TRPTMP = SQRT(4.46911-2* ALOG( UNIF( IX)))
      IF ( TRPTMP* UNIF( IX) .GT. 2.11403) GO TO 180
      GO TO 340
  210 IF ( U .LT. .949991) GO TO 260
C
C  AREA C
C
  230 TRPTMP = 1.8404+ UNIF( IX)*.273629
      IF (.398942* EXP(- TRPTMP* TRPTMP/2)-.443299 +
     X TRPTMP*.209694 .LT. UNIF( IX)*  4.27026E-02) GO TO 230
      GO TO 340
  260 IF ( U .LT. .925852) GO TO 310
C
C  AREA D
C
  280 TRPTMP = .28973+ UNIF( IX)*1.55067
      IF (.398942* EXP(- TRPTMP* TRPTMP/2)-.443299 +
     X TRPTMP*.209694 .LT. UNIF( IX)* 1.59745E-02) GO TO 280
      GO TO 340
  310 TRPTMP = UNIF( IX)*.28973
C
C  AREA E
C
      IF (.398942* EXP( TRPTMP* TRPTMP/2)-.382545
     X .LT. UNIF( IX)*1.63977E-02)  GO TO 310
  340 IF ( UO .GT. .5) GO TO 370
      TRPTMP = - TRPTMP
  370 TRPNRM = TRPTMP
      RETURN
      END
```

Figure L.19. The function TRPNRM.

```
      FUNCTION UNIF( IX)
C
C  PORTABLE RANDOM NUMBER GENERATOR USING THE RECURSION:
C  IX = 16807 * IX MOD (2**(31) - 1)
C  USING ONLY 32 BITS, INCLUDING SIGN.
C  SOME COMPILERS REQUIRE THE DECLARATION:
C  INTEGER*4 IX, K1
C
C  INPUT:
C    IX = INTEGER GREATER THAN 0 AND LESS THAN 2147483647
C
C  OUTPUTS:
C    IX = NEW PSEUDORANDOM VALUE,
C    UNIF = A UNIFORM FRACTION BETWEEN 0 AND 1.
C
       K1 = IX/127773
       IX = 16807*( IX - K1*127773) - K1 * 2836
       IF ( IX .LT. 0) IX = IX + 2147483647
       UNIF = IX*4.656612875E-10
       RETURN
       END
```

Figure L.20. The function UNIF.

```
      FUNCTION VBETA( PHI, A, B)
C
C  INVERSE OF THE BETA CDF
C  REF.: ABRAMOVITZ, M. AND STEGUN, I., HANDBOOK OF MATHEMATICAL
C  FUNCTIONS WITH FORMULAS, GRAPHS, AND MATHEMATICAL TABLES,
C  NAT. BUR. STANDARDS, GOVT. PRINTING OFF.
C
C  INPUTS:
C    PHI = PROBABILITY TO BE INVERTED, 0 < PHI < 1.
C    A, B = THE 2 SHAPE PARAMETERS OF THE BETA
C
C  OUTPUTS:
C    VBETA = INVERSE OF THE BETA CDF
C
C  AUXILIARY ROUTINE:
C    VNORM
C
C  ACCURACY: ABOUT 1 DECIMAL DIGIT
C  EXCEPT IF A OR B =1., IN WHICH CASE ACCURACY= MACHINE PRECISION.
C
      DATA DWARF /.1E-9/
C
      IF ( PHI .GT. DWARF) GO TO 100
      VBETA = 0.
      RETURN
  100 IF ( PHI .LT. (1. - DWARF)) GO TO 150
      VBETA = 1.
      RETURN
  150 IF ( B .NE. 1.) GO TO 200
      VBETA = PHI**(1./A)
      RETURN
  200 IF ( A .NE. 1.) GO TO 300
      VBETA = 1. - (1. - PHI)**(1./B)
      RETURN
  300 YP = - VNORM( PHI, IFAULT)
      GL = ( YP* YP - 3.)/ 6.
      AD = 1./( 2.* A - 1.)
      BD = 1./( 2.* B - 1.)
      H = 2./( AD + BD)
      W =( YP* SQRT( H+ GL)/ H)-
     X( BD- AD)*( GL+.83333333-.66666666/H)
      VBETA = A/( A + B* EXP(2.* W))
      RETURN
      END
```

Figure L.21. The function VBETA.

```
      FUNCTION VCAUCH( PHI)
C
C  INVERSE OF STANDARD CAUCHY DISTRIBUTION CDF.
C
C  MACHINE EPSILON
C
      DATA DWARF/ .1E-30/
C
      ARG =  3.1415926*( 1. - PHI)
      SINARG = AMAX1( DWARF, SIN( ARG))
      VCAUCH = COS( ARG)/ SINARG
      RETURN
      END
```

Figure L.22. The function VCAUCH.

```
      FUNCTION VCHI2( PHI, V, G, IFAULT)
C
C  EVALUATE THE PERCENTAGE POINTS OF THE CHI-SQUARED
C  PROBABILITY DISTRIBUTION FUNCTION.
C  SLOW BUT ACCURATE INVERSION OF CHI-SQUARED CDF.
C
C  BASED ON ALGORITHM AS 91 APPL. STATIST. (1975) VOL.24, NO.3
C
C  INPUTS:
C    PHI = PROBABILITY TO BE INVERTED.
C    PHI SHOULD LIE IN THE RANGE 0.000002 TO 0.999998,
C    V = DEGREES OF FREEDOM, AND MUST BE POSITIVE,
C    G MUST BE SUPPLIED AND SHOULD BE EQUAL TO LN(GAMMA(V/2.0))
C    E.G. USING FUNCTION ALOGAM.
C
C  OUTPUT:
C    VCHI2 = INVERSE OF  CHI-SQUARE CDF OF PHI
C    IFAULT = 4 IF INPUT ERROR, ELSE 0
C
C  AUXILIARY ROUTINES:
C    VNORM, GAMAIN
C
C  DESIRED ACCURACY AND LN( 2.):
C
      DATA E, AA/0.5E-5, 0.6931471805E0/
C
C  CHECK FOR UNUSUAL INPUT
C
      IF( V .LE. 0.) GO TO 9000
      IF ( PHI .LE. .000002 .OR. PHI .GE. .999998 ) GO TO 9000
      XX = 0.5 * V
      C = XX - 1.0
C
C  STARTING APPROXIMATION FOR SMALL CHI-SQUARED
C
      IF (V .GE. -1.24 * ALOG( PHI)) GOTO 1
      CH = ( PHI * XX * EXP(G + XX * AA)) ** (1.0 / XX)
      IF (CH - E) 6, 4, 4
C
C  STARTING APPROXIMATION FOR V LESS THAN OR EQUAL TO 0.32
C
    1 IF (V .GT. 0.32) GOTO 3
      CH = 0.4
      A = ALOG(1.0 - PHI)
    2 Q = CH
      P1 = 1.0 + CH * (4.67 + CH)
      P2 = CH * (6.73 + CH * (6.66 + CH))
      T = -0.5 + (4.67 + 2.0 * CH) / P1 -
     X   (6.73 + CH * (13.32 + 3.0 * CH)) / P2
      CH = CH - (1.0 - EXP(A + G + 0.5 * CH + C * AA) * P2 / P1) / T
      IF (ABS(Q / CH - 1.0) - 0.01) 4, 4, 2
C
C  GET THE CORRESPONDING NORMAL DEVIATE:
C
    3 X = VNORM( PHI, IFAULT)
C
C  STARTING APPROXIMATION USING WILSON AND HILFERTY ESTIMATE
C
      P1 = 0.222222 / V
      CH = V * (X * SQRT(P1) + 1.0 - P1) ** 3
C
C  STARTING APPROXIMATION FOR P TENDING TO 1
```

(*continued*)

```
C
      IF (CH .GT. 2.2 * V + 6.0)
     X   CH = -2.0 * (ALOG(1.0 - PHI) - C * ALOG(0.5 * CH) + G)
C
C  CALL TO ALGORITHM AS 32 AND CALCULATION OF SEVEN TERM
C  TAYLOR SERIES
C
    4 Q = CH
      P1 = 0.5 * CH
      P2 = PHI - GAMAIN(P1,XX,G,IF1)
      IF (IF1 .EQ. 0) GOTO 5
      IFAULT = 4
      RETURN
    5 T = P2 * EXP(XX * AA + G + P1 - C * ALOG(CH))
      B = T / CH
      A = 0.5 * T - B * C
      S1 = (210.0+A*(140.0+A*(105.0+A*(84.0+A*(70.0+60.0*A))))) / 420.0
      S2 = (420.0+A*(735.0+A*(966.0+A*(1141.0+1278.0*A)))) / 2520.0
      S3 = (210.0 + A * (462.0 + A * (707.0 + 932.0 * A))) / 2520.0
      S4 =(252.0+A*(672.0+1182.0*A)+C*(294.0+A*(889.0+1740.0*A)))/5040.
      S5 = (84.0 + 264.0 * A + C * (175.0 + 606.0 * A)) / 2520.0
      S6 = (120.0 + C * (346.0 + 127.0 * C)) / 5040.0
      CH = CH+T*(1.0+0.5*T*S1-B*C*(S1-B*(S2-B*(S3-B*(S4-B*(S5-B*S6))))))
      IF (ABS(Q / CH - 1.0) .GT. E) GOTO 4
C
    6 VCHI2 = CH
      RETURN
 9000 IFAULT = 4
C
C  TRY TO GIVE SENSIBLE RESPONSE
C
      VCHI2 = 0.
      IF (  PHI .GT. .999998) VCHI2 = V + 4.*SQRT(2.*V)
      RETURN
      END
```

Figure L.23. The function VCHI2.

```
      FUNCTION VCHISQ( PHI, DF)
C
C  INVERSE OF THE CHI-SQUARE C.D.F.
C  TO GET INVERSE CDF OF K-ERLANG SIMPLY USE:
C  .5* VCHISQ( PHI, 2.*K);
C  WARNING: THE METHOD IS ONLY ACCURATE TO ABOUT 2 PLACES FOR
C  DF < 5 BUT NOT EQUAL TO 1. OR 2.
C
C  REF.:  ABRAMOVITZ AND STEGUN
C
C  INPUTS:
C    PHI = PROBABILITY
C    DF= DEGREES OF FREEDOM, NOTE, THIS IS REAL
C
C  OUTPUTS:
C    VCHISQ = F INVERSE OF PHI.
C
C    SQRT(.5)
C
      DATA SQP5 /.7071067811E0/
      DATA DWARF /.1E-15/
      IF ( DF .NE. 1.) GO TO 100
C
C  WHEN DF=1 WE CAN COMPUTE IT QUITE ACCURATELY
C  BASED ON FACT IT IS A SQUARED STANDARD NORMAL
C
      Z = VNORM( 1. -(1. - PHI)/2., IFAULT)
      VCHISQ = Z* Z
      RETURN
  100 IF ( DF .NE. 2.) GO TO 200
C
C  WHEN DF = 2 IT CORRESPONDS TO EXPONENTIAL WITH MEAN 2.
C
      ARG = 1. - PHI
      IF ( ARG .LT. DWARF) ARG = DWARF
      VCHISQ = - ALOG( ARG)* 2.
      RETURN
  200 IF ( PHI .GT. DWARF) GO TO 300
      VCHISQ = 0.
      RETURN
  300 Z = VNORM( PHI, IFAULT)
      SQDF = SQRT(DF)
      ZSQ = Z* Z
      CH =-(((3753.*ZSQ+4353.)*ZSQ-289517.)*ZSQ-289717.)*
     X Z*SQP5/9185400.
      CH=CH/SQDF+(((12.*ZSQ-243.)*ZSQ-923.)*ZSQ+1472.)/25515.
      CH=CH/SQDF+((9.*ZSQ+256.)*ZSQ-433.)*Z*SQP5/4860.
      CH=CH/SQDF-((6.*ZSQ+14.)*ZSQ-32.)/405.
      CH=CH/SQDF+(ZSQ-7.)*Z*SQP5/9.
      CH=CH/SQDF+2.*(ZSQ-1.)/3.
      CH=CH/SQDF+Z/SQP5
      VCHISQ = DF*( CH/SQDF + 1.)
      RETURN
      END
```

Figure L.24. The function VCHISQ.

```
      FUNCTION VF( PHI, DFN, DFD, W)
      DIMENSION W(4)
C
C  INVERSE OF THE F CDF
C  REF.: ABRAMOVITZ AND STEGUN
C
C  INPUTS:
C    PHI = PROBABILITY TO BE INVERTED.  0. < PHI < 1.
C    DFN= DEGREES OF FREEDOM OF THE NUMERATOR
C    DFD = DEGREES OF FREEDOM OF THE DENOMINATOR
C      OF THE F DISTRIBUTION.
C    W = WORK VECTOR OF LENGTH 4. W(1) MUST BE SET TO -1 ON ALL
C      INITIAL CALLS TO VF WHENEVER DFN = 1 OR DFD = 1. AS LONG AS
C      DFN AND DFD DO NOT CHANGE, W SHOULD NOT BE ALTERED. IF,
C      HOWEVER, DFN OR DFD SHOULD CHANGE, W(1) MUST BE RESET TO -1.
C
C  OUTPUTS:
C    VF = INVERSE OF F CDF EVALUATED AT PHI.
C    W  = WORK VECTOR OF LENGTH 4. W RETAINS INFORMATION BETWEEN
C      SUCCESSIVE CALLS TO VF WHENEVER DFN OR DFD EQUAL 1.
C
C  ACCURACY: ABOUT 1 DECIMAL DIGIT, EXCEPT
C  IF DFN OR DFD =1. IN WHICH CASE ACCURACY = ACCURACY OF VSTUD.
C  ELSE ACCURACY IS THAT OF VBETA.
C
      DATA DWARF /.1E-15/
C
C  AUXILIARY ROUTINES:
C    VBETA, VSTUD
C
      IF ( DFN .NE. 1.) GO TO 200
      FV = VSTUD(( 1. + PHI)/2., DFD, W, IFAULT)
      VF = FV* FV
      RETURN
C
  200 IF ( DFD .NE. 1.) GO TO 300
      FV = VSTUD(( 2. - PHI)/2., DFN, W, IFAULT)
      IF ( FV .LT. DWARF) FV = DWARF
      VF = 1./(FV* FV)
      RETURN
C
  300 FV = 1. - VBETA( PHI, DFN/2., DFD/2.)
      IF ( FV .LT. DWARF) FV = DWARF
      VF = (DFD/DFN)*( 1. - FV)/( FV)
      RETURN
      END
```

Figure L.25. The function VF.

```
      FUNCTION VNHOMO( U, N, RATE, T, R, K, IFAULT)
      REAL RATE( N), T( N)
C
C  INVERSE OF THE DYNAMIC EXPONENTIAL,
C  A.K.A. NONHOMOGENOUS POISSON
C
C  INPUTS:
C    U = PROBABILITY TO BE INVERTED
C    N = NO. INTERVALS IN A CYCLE
C    RATE(K) = ARRIVAL RATE IN INTERVAL K OF CYCLE
C    T(K) = END POINT OF INTERVAL K, SO T(N) = CYCLE LENGTH
C    R = CURRENT TIME MOD CYCLE LENGTH, E.G. 0 AT TIME 0
C    K = CURRENT INTERVAL OF CYCLE, E.G. 1 AT TIME 0
C    NOTE: R AND K WILL BE UPDATED FOR THE NEXT CALL
C
C  OUTPUTS:
C    VNHOMO = INTERARRIVAL TIME
C    R = TIME OF NEXT ARRIVAL MOD CYCLE LENGTH
C    K = INTERVAL IN WHICH NEXT ARRIVAL WILL OCCUR
C    IFAULT = 5 IF ERROR IN INPUT, ELSE 0
C
      IFAULT = 0
      X = 0.
C
C  GENERATE INTERARRIVAL TIME WITH MEAN 1
C
      E = - ALOG(1. - U)
C
C  SCALED TO CURRENT RATE DOES IT FALL IN CURRENT INTERVAL?
C
  100 STEP = T(K) - R
      IF ( STEP .LT. 0.) GO TO 900
      RNOW = RATE( K)
      IF ( RNOW .GT. 0.) GO TO 110
      IF ( RNOW .EQ. 0.) GO TO 120
      GO TO 900
C
C  INTERARRIVAL TIME IF AT CURRENT RATE
C
  110 U = E/ RNOW
      IF ( U .LE. STEP) GO TO 300
C
C  NO, JUMP TO END OF THIS INTERVAL
C
  120 X = X + STEP
      E = E - STEP* RNOW
      IF ( K .EQ. N) GO TO 200
      R = T(K)
      K = K+ 1
      GO TO 100
  200 R = 0.
      K = 1
      GO TO 100
C
C  YES, WE STAYED IN INTERVAL K
C
  300 VNHOMO = X + U
      R = R + U
      RETURN
  900 IFAULT = 5
      RETURN
      END
```

Figure L.26. The function VNHOMO.

```
      FUNCTION VNORM( PHI, IFAULT)
C
C  VNORM RETURNS THE INVERSE OF THE CDF OF THE NORMAL  DISTRIBUTION.
C  IT USES A RATIONAL APPROXIMATION WHICH SEEMS TO HAVE A RELATIVE
C  ACCURACY OF ABOUT 5 DECIMAL PLACES.
C  REF.: KENNEDY AND GENTLE, STATISTICAL COMPUTING, DEKKER, 1980.
C
C  INPUTS:
C    PHI = PROBABILITY, 0 <= PHI <= 1.
C
C  OUTPUTS:
C    F INVERSE OF PHI, I.E., A VALUE SUCH THAT
C    PROB( X <= VNORM) = PHI.
C    IFAULT = 6 IF PHI OUT OF RANGE, ELSE 0
C
      DATA PLIM /1.0E-18/
      DATA P0/-0.322232431088E0/, P1 / -1.0/, P2 / -0.342242088547E0/
      DATA  P3 / -0.0204231210245E0/,    P4/-0.453642210148E-4/
      DATA Q0/0.099348462606E0/, Q1/0.588581570495E0/
      DATA Q2/0.531103462366E0/,  Q3/0.10353775285E0/
      DATA Q4/0.38560700634E-2/
C
      IFAULT = 0
      P = PHI
      IF (P.GT.0.5) P = 1. - P
      IF ( P .GE. PLIM) GO TO 100
C
C  THIS IS AS FAR OUT IN THE TAILS AS WE GO
C
      VTEMP =  8.
C
C  CHECK FOR INPUT ERROR
C
      IF ( P .LT. 0.) GO TO 9000
      GO TO 200
  100 Y = SQRT(-ALOG(P*P))
      VTEMP = Y + ((((Y*P4 + P3)*Y + P2)*Y + P1)*Y + P0)/
     X     ((((Y*Q4 + Q3)*Y + Q2)*Y + Q1)*Y + Q0)
  200 IF ( PHI .LT. 0.5) VTEMP = - VTEMP
      VNORM = VTEMP
      RETURN
 9000 IFAULT = 6
      RETURN
      END
```

Figure L.27. The function VNORM.

```
      FUNCTION VSTUD( PHI, DF, W, IFAULT)
      DIMENSION W(4)
C
C  GENERATE THE INVERSE CDF OF THE STUDENT T
C.
C  INPUTS:
C    PHI = PROBABILITY , 0 <= PHI <= 1
C    DF = DEGREES OF FREEDOM
C    W = WORK VECTOR OF LENGTH 4 WHICH DEPENDS ONLY ON DF.
C      VSTUD WILL RECALCULATE,  E.G., ON FIRST CALL, IF
C      W(1) < 0. ON INPUT.
C
C  OUTPUTS:
C    VSTUD = F INVERSE OF PHI, I.E., VALUE SUCH THAT
C      PROB( X <= VSTUD) = PHI.
C    W = WORK VECTOR OF LENGTH 4. W RETAINS INFORMATION BETWEEN CALLS
C          WHEN DF DOES NOT CHANGE.
C    IFAULT = 7 IF INPUT ERROR, ELSE 0.
C
C  AUXILIARY ROUTINE:
C    VNORM
C
C  ACCURACY IN DECIMAL DIGITS OF ABOUT:
C    5 IF DF >= 8.,
C    MACHINE PRECISION IF DF = 1. OR 2.,
C    3 OTHERWISE IF DF < 8.
C
C  REFERENCE: G. W. HILL, COMMUNICATIONS OF THE ACM, VOL. 13, NO. 10,
C  OCT. 1970.
C
C  MACHINE EPSILON
C
      DATA DWARF/ .1E-30/
C
C  PI OVER 2
C
      DATA PIOVR2 /1.5707963268E0/
C
C  CHECK FOR INPUT ERROR
C
      IFAULT = 0
      IF (  PHI .LT. 0. .OR. PHI .GT. 1.) GO TO 9000
      IF ( DF .LE. 0.) GO TO 9000
C
C  IF IT IS A CAUCHY USE SPECIAL METHOD
C
      IF ( DF .EQ. 1.) GO TO 300
C
C  THERE IS ALSO A SPECIAL, EXACT METHOD IF DF = 2
C
      IF ( DF .EQ. 2.) GO TO 400
C
C  CHECK TO SEE IF WE'RE NOT TOO FAR OUT IN THE TAILS.
C
      IF (PHI.GT.DWARF .AND. (1. - PHI).GT.DWARF) GO TO 205
         T = 10.E30
         GO TO 290
```

(*continued*)

```
C
C  GENERAL CASE
C
C  FIRST, CONVERT THE CUMULATIVE PROBABILITY TO THE TWO-TAILED
C  EQUIVALENT USED BY THIS METHOD.
C
  205 PHIX2 = 2. * PHI
      P2TAIL = AMIN1(PHIX2, 2.0 - PHIX2)
C
C  BEGIN THE APROXIMATION
C
C  IF W(1) >= 0 THEN SKIP THE COMPUTATIONS THAT DEPEND SOLELY ON DF.
C    THE VALUES FOR THESE VARIABLES WERE STORED IN W DURING A PREVIOUS
C    CALL.
C
      IF (W(1) .GT. 0.) GO TO 250
          W(2) = 1.0/(DF - 0.5)
          W(1) = 48.0/(W(2) * W(2))
          W(3) = ((20700.0*W(2)/W(1)-98.)*W(2)-16.)*W(2) + 96.36
          W(4)=((94.5/(W(1)+W(3))-3.)/W(1) + 1.)*SQRT(W(2)*PIOVR2)*DF
  250 X = W(4) * P2TAIL
      C = W(3)
      Y = X ** (2./DF)
      IF (Y.LE.(.05 + W(2))) GO TO 270
C
C  ASYMPTOTIC INVERSE EXPANSION ABOUT NORMAL
C
          X = VNORM( P2TAIL * 0.5, IFAULT)
          Y = X * X
          IF (DF.LT.5) C = C + 0.3 * (DF - 4.5) * (X + .6)
          C = (((.05*W(4)*X-5.)*X-7.)*X-2.)*X+W(1)+C
          Y = (((((.4*Y+6.3)*Y+36.)*Y+94.5)/C-Y-3.)/W(1)+1.)*X
          Y = W(2) * Y * Y
          IF (Y .LE. 0.002) GO TO 260
              Y = EXP(Y) - 1.
              GO TO 280
  260         Y = .5 * Y * Y + Y
              GO TO 280
C
  270     Y2 = (DF + 6.)/(DF + Y) - 0.089 * W(4) - .822
          Y2 = Y2 * (DF + 2.) * 3.
          Y2 = (1./Y2 + .5/(DF + 4.)) * Y - 1.
          Y2 = Y2 * (DF + 1.)/(DF + 2.)
          Y  = Y2 + 1./Y
C
  280 T = SQRT(DF * Y)
  290 IF (PHI.LT.0.5) T = - T
      VSTUD = T
      RETURN
C
C  INVERSE OF STANDARD CAUCHY DISTRIBUTION CDF.
C  RELATIVE ACCURACY = MACHINE PRECISION, E.G. 5 DIGITS
C
  300 ARG =  3.1415926*( 1. - PHI)
      SINARG = AMAX1( DWARF, SIN( ARG))
      VSTUD = COS( ARG)/ SINARG
      RETURN
```

(continued)

```
C
C  SPECIAL CASE OF DF = 2.
C  RELATIVE ACCURACY = MACHINE PRECISION, E.G., 5 DIGITS
C
  400 IF  (  PHI .GT. .5) GO TO 440
      T = 2.* PHI
      GO TO 450
  440 T = 2.*( 1. - PHI)
  450 IF ( T .LE. DWARF) T = DWARF
      VTEMP = SQRT((2./(T*( 2. - T))) - 2.)
      IF ( PHI .LE. .5) VTEMP = - VTEMP
      VSTUD = VTEMP
      RETURN
 9000 IFAULT = 7
      RETURN
      END
```

Figure L.28. The function VSTUD.

```
      SUBROUTINE BINSRH( U, K, N, CDF)
      REAL CDF(N)
C
C  FIND SMALLEST K SUCH THAT CDF( K) >= U BY BINARY SEARCH
C
C  INPUTS:
C    U = PROBABILITY
C    CDF = A CDF FOR A VARIABLE DEFINED ON (1, 2,..., N)
C    N = DIMENSION OF CDF
C
      K = 1
      IF ( U .LE. CDF(1)) GO TO 300
      MAXI = N
      MINI = 2
C
  100 K = ( MAXI + MINI)/ 2
      IF ( U .GT. CDF(K)) GO TO 200
      IF ( U .GT. CDF( K - 1)) GO TO 300
      MAXI = K - 1
      GO TO 100
  200 MINI = K + 1
      GO TO 100
  300 RETURN
      END
```

Figure L.29. The subroutine BINSRH.

```
      SUBROUTINE FNORM( X ,PHI, UNL)
C
C  EVALUATE THE STANDARD NORMAL CDF AND THE UNIT LINEAR LOSS
C
C  INPUT:
C    X = A REAL NUMBER
C
C  OUTPUTS:
C    PHI = AREA UNDER UNIT NORMAL CURVE TO THE LEFT OF X.
C    UNL = E(MAX(0,X-Z)) WHERE Z IS UNIT NORMAL,
C    I.E., UNL = UNIT NORMAL LOSS FUNCTION =
C    INTEGRAL FROM MINUS INFINITY TO X OF ( X - Z) F( Z) DZ, WHERE
C    F(Z) IS THE UNIT NORMAL DENSITY.
C
C  WARNING: RESULTS ARE ACCURATE ONLY TO ABOUT 5 PLACES
C
      Z = X
      IF ( Z .LT.  0.) Z = -Z
      T = 1./(1. + .2316419 * Z)
      PHI = T*(.31938153 + T*(-.356563782 + T*(1.781477937 +
     X  T*( -1.821255978 + T*1.330274429))))
      E2 = 0.
C
C  6 S. D. OUT GETS TREATED AS INFINITY
C
      IF ( Z .LE. 6.) E2 = EXP(-Z*Z/2.)*.3989422803
      PHI = 1. - E2* PHI
      UNL = Z* PHI + E2
      IF ( X .GE. 0.) RETURN
      PHI = 1. - PHI
      UNL = UNL + X
      RETURN
      END
```

Figure L.30. The subroutine FNORM.

```
      SUBROUTINE TBINOM( N, P, CDF, IFAULT)
      REAL CDF( N)
C
C  TABULATE THE BINOMIAL CDF
C
C  INPUTS:
C    N = NO. OF POSSIBLE OUTCOMES
C    (SHIFTED BY 1 SO THEY ARE 1,2...N)
C    P = SECOND PARAMETER OF BINOMIAL, 0 <= P <= 1.
C    E.G., A BINOMIAL WITH 10 TRIALS AND A PROBABILITY OF
C    SUCCESS OF .2 AT EACH TRIAL HAS 11 POSSIBLE
C    OUTCOMES,  THUS TBINOM IS CALLED WITH N = 11 AND P = .2
C
C  OUTPUTS:
C    CDF( I) = PROB( X <= I-1)
C      = PROBABILITY OF LESS THAN I SUCCESSES
C    IFAULT = 8 IF INPUT ERROR, ELSE 0
C
C  SOMEWHAT CIRCUITOUS METHOD IS USED TO AVOID UNDERFLOW
C  AND OVERFLOW WHICH MIGHT OTHERWISE OCCUR FOR N > 120.
C
C  TOLERANCE FOR SMALL VALUES
      DATA TOLR /1.E-8/
C
C  CHECK FOR BAD INPUT
C
      IF (( N .LE. 0).OR.( P .LT. 0.).OR.( P .GT. 1.)) GO TO 9000
C
C  WATCH OUT FOR BOUNDARY CASES OF P
C
      IF ( P .GT. 0.) GO TO 50
      DO 30 J = 1,N
      CDF(J) = 1.
   30 CONTINUE
      RETURN
   50 IF ( P .LT. 1.) GO TO 70
      DO 60 J = 1, N
      CDF(J) = 0.
   60 CONTINUE
      CDF( N) = 1.
      RETURN
C
C  TYPICAL VALUES OF P START HERE
C
   70   QOP = ( 1. - P)/ P
      POQ = P/( 1. - P)
      R = 1.
C
C  SET K APPROXIMATELY = MEAN
C
      K = N* P + 1.
      IF ( K .GT. N) K = N
      CDF(K) = 1.
      K1 = K-1
      TOT = 1.
      IF ( K1 .LE. 0) GO TO 150
      J = K
C
C  CALCULATE P.M.F. PROBABILITIES RELATIVE TO P(K).
C  FIRST BELOW K
C
      DO 100 I = 1, K1
      J = J - 1
```

(continued)

```
C
C  RELATIVELY SMALL PROBABILITIES GET SET TO 0.
C
      IF ( R .LE. TOLR) R = 0.
      R = R* QOP *J/( N - J)
      CDF(J) = R
      TOT = TOT + R
  100 CONTINUE
C
C  NOW ABOVE K
C
  150 IF ( K .EQ. N) GO TO 300
      R = 1.
      K1 = K + 1
      DO 200 J = K1 , N
C
C  RELATIVELY SMALL PROBABILITIES GET SET TO 0.
C
      IF ( R .LE. TOLR) R = 0.
      R = R * POQ*( N - J + 1)/( J -1)
      TOT = TOT + R
      CDF( J ) = R
  200 CONTINUE
C
C  NOW RESCALE AND CONVERT P.M.F. TO C.D.F.
C
  300 RUN = 0.
      DO 400 J = 1, N
      RUN = RUN + CDF( J)/ TOT
      CDF( J) = RUN
  400 CONTINUE
      RETURN
 9000 IFAULT = 8
      RETURN
      END
```

Figure L.31. The subroutine TBINOM.

```
      SUBROUTINE TPOISN( N, P, CDF, IFAULT)
      REAL CDF(N)
C
C  TABULATE THE CDF OF A TRUNCATED POISSON WITH PARAMETER P.
C  N VALUES ARE TABULATED.
C  SOMEWHAT CIRCUITOUS SCALING METHOD IS USED TO AVOID
C  UNDERFLOW,  OVERFLOW, AND USE OF THE EXP FUNCTION.
C
C  OUTPUTS:
C    CDF( I) = THE CDF OF THE POISON DISTRIBUTION
C    SHIFTED BY 1 SO THAT CDF( I) = PROB( X <= I-1).
C    IFAULT = 9 IF INPUT ERROR, ELSE 0
C
C  TOLERANCE FOR SMALL VALUES
C
      DATA TOLR/.1E-7/
C
C  CHECK FOR BAD INPUT
C
      IFAULT = 0
      IF (( N .LE. 0).OR.( P .LT. 0.)) GO TO 9000
C
C  WATCH OUT FOR BOUNDARY CASES OF P
C
      IF ( P .GT. 0.) GO TO 50
      DO 30 J = 1, N
      CDF( J) = 1.
   30 CONTINUE
      RETURN
C
C  TYPICAL VALUES OF P START HERE
C
   50 R = 1.
      K = P + 1.
      IF ( K .GT. N) K = N
      CDF(K) = 1.
      K1 = K-1
      TOT = 1.
      IF ( K1 .LE. 0) GO TO 150
      J = K
C
C  CALCULATE P.M.F. PROBABILITIES RELATIVE TO P(K)
C
      DO 100 I = 1, K1
C
C  RELATIVELY SMALL PROBABILITIES GET SET TO 0.
C
      IF ( R .LE. TOLR) R = 0.
      J = J - 1
      R = R*J/P
      CDF(J) = R
      TOT = TOT + R
  100 CONTINUE
C
C  NOW GO UP
C
  150 IF ( K .EQ. N) GO TO 300
      R = 1.
      K1 = K + 1
      DO 200 J = K1 , N
```

(continued)

```
C
C  RELATIVELY SMALL PROBABILITIES GET SET TO 0.
C
      IF ( R .LE. TOLR) R = 0.
      R = R* P/( J - 1)
      TOT = TOT + R
      CDF( J ) = R
  200 CONTINUE
C
C  NOW RESCALE AND CONVERT TO C.D.F.
C
  300 RUN = 0.
      DO 400 J = 1, N
      RUN = RUN + CDF( J)/ TOT
      CDF( J) = RUN
  400 CONTINUE
      RETURN
C
 9000 IFAULT = 9
      RETURN
      END
```

Figure L.32. The subroutine TPOISN.

APPENDIX X
Examples of Simulation Programming

This appendix contains most of the programs and examples referred to in Chapters 7 and 8. A number of comments are in order.

First and foremost, these programs are illustrations: they are not intended to be copied slavishly. The Fortran programs, for example, are written in the current ANSI standard Fortran [ANSI (1978)], using the full language. They will not run on machines still using Fortran IV, nor on compilers which accept only the official ANSI subset. Simula programs are presented with keywords underlined and in lower case, since we believe this is easier to read than the all-upper-case representation imposed by some machines. All the programs presented have been tested in one form or another, though not necessarily with exactly the text given in this appendix.

While writing this book we used a variety of machines. Most of the numerical results in Chapter 8 were obtained using a CDC Cyber 173; almost all of them were checked on a XDS Sigma 6. The examples in Simula and GPSS were also run on a Cyber 173. The examples in Simscript were run on an IBM 4341. Some of the GPSS programs were also checked on this machine.

The reader should be aware that if he *does* run our programs on a different machine, he may get different answers despite using the same portable random number generator. Such functions as ERLANG and NXTARR use floating-point arithmetic; NXTARR in particular cumulates floating-point values. A very minor difference in floating-point representations on two different machines may be just enough to throw two supposedly identical simulations out of synchronization: once this happens, all further resemblance between their outputs may be lost.

```
      PROGRAM BOMBER
      COMMON /TIMCOM/TIME
      INTEGER INRNGE, BOMB, READY, BURST, ENDSIM
      INTEGER SEED, EVENT, I, N, GOUT, PDOWN
      REAL TIME, DIST, TOFFLT, NORMAL, UNIF, R
      LOGICAL PLANOK(5), GUNOK(3), GUNRDY(3), TRACE
      PARAMETER (INRNGE = 1, BOMB = 2, READY = 3,
     X           BURST = 4, ENDSIM = 5)
      EXTERNAL PUT, GET, INIT, UNIF, NORMAL, DIST, TOFFLT
C
C  EXAMPLE 7.1.1 FROM SECTION 7.1.4
C
C  TIME IS MEASURED IN SECONDS.
C  TIME 0 IS WHEN THE FIRST AIRCRAFT
C  COMES INTO RANGE.
C
C  CHANGE THE FOLLOWING TO SUPPRESS TRACE
C
      TRACE = .TRUE.
C
C  INITIALIZE - ALL PLANES ARE OUT OF RANGE, ALL GUNS ARE
C  INTACT AND READY TO FIRE.
C  NO PLANES HAVE BEEN SHOT DOWN.
C
      CALL INIT
      DO 1 I=1,5
    1 PLANOK(I) = .FALSE.
      PDOWN = 0
      DO 2 I=1,3
      GUNOK(I) = .TRUE.
    2 GUNRDY(I) = .TRUE.
      GOUT = 0
      SEED = 12345
C
C  R IS THE RANGE FOR OPENING FIRE.
C  THE SCHEDULE INITIALLY CONTAINS JUST TWO EVENTS: PLANE 1
C  COMES INTO RANGE AND THE END OF THE SIMULATION.
C
      R = 3000.0
      CALL PUT(INRNGE, (3900.0-R)/300.0, 1)
      CALL PUT(ENDSIM, 50.0, 0)
   99 CALL GET(EVENT, N)
      IF (EVENT .EQ. INRNGE) GO TO 100
      IF (EVENT .EQ. BOMB)   GO TO 200
      IF (EVENT .EQ. READY)  GO TO 300
      IF (EVENT .EQ. BURST)  GO TO 400
      IF (EVENT .EQ. ENDSIM) GO TO 500
C
C  THIS SECTION SIMULATES AN AIRCRAFT COMING INTO RANGE
C
C  IF IT IS NOT THE LAST AIRCRAFT, SCHEDULE ITS SUCCESSOR.
C  IF THERE IS A GUN READY TO FIRE, IT WILL DO SO.
C  WHEN A GUN FIRES, WE MUST SCHEDULE THE BURST AND ALSO
C  THE TIME WHEN THE GUN IS RELOADED.
C  FINALLY, SCHEDULE AN ARRIVAL AT THE BOMBING POINT.
```

(*continued*)

```
C
  100 IF (TRACE) PRINT 910,TIME,N
  910 FORMAT (F6.2,'   AIRCRAFT',I4,' IN RANGE')
      IF (N .LT. 5) CALL PUT(INRNGE, TIME+2.0, N+1)
      PLANOK(N) = .TRUE.
      DO 101 I=1,3
      IF (.NOT. GUNOK(I) .OR. .NOT. GUNRDY(I)) GO TO 101
      IF (TRACE) PRINT 911,TIME,I,N
  911 FORMAT (F6.2,'   GUN',I4,' FIRES AT AIRCRAFT',I4)
      CALL PUT(BURST, TIME+TOFFLT(N), N)
      CALL PUT(READY, TIME+NORMAL(5.0, 0.5, SEED), I)
      GUNRDY(I) = .FALSE.
  101 CONTINUE
      CALL PUT(BOMB, TIME+R/300.0, N)
      GO TO 99
C
C  THIS SECTION SIMULATES AN AIRCRAFT DROPPING ITS BOMBS
C
C  IF THE PLANE HAS BEEN DESTROYED, THE EVENT IS IGNORED.
C  OTHERWISE, WE DECIDE FOR EACH GUN IN TURN IF IT IS
C  HIT OR NOT.
C
  200 IF (PLANOK(N)) THEN
        IF (TRACE) PRINT 920,TIME,N
  920   FORMAT (F6.2,'   AIRCRAFT',I4,' DROPS ITS BOMBS')
        DO 201 I=1,3
        IF ( (UNIF(SEED) .LE. 0.2) .AND. GUNOK(I) ) THEN
          GUNOK(I) = .FALSE.
          GOUT = GOUT + 1
          IF (TRACE) PRINT 921,TIME,I
  921     FORMAT (F6.2,'   GUN',I4,' DESTROYED')
        ENDIF
  201   CONTINUE
      ENDIF
      GO TO 99
C
C  THIS SECTION SIMULATES THE END OF LOADING A GUN
C
C  IF THE GUN HAS BEEN DESTROYED, THE EVENT IS IGNORED.
C  OTHERWISE, LOOK FOR A TARGET: WE TRY FIRST THE NEAREST
C  ONCOMING PLANE, THEN THE NEAREST RECEDING PLANE.  IF
C  A TARGET IS FOUND, SCHEDULE THE BURST AND THE END OF
C  RELOADING; OTHERWISE, MARK THE GUN AS READY.
C
  300 IF (GUNOK(N)) THEN
        DO 301 I=1,5
        IF (PLANOK(I) .AND. DIST(I) .LE. 0) GO TO 303
  301   CONTINUE
        DO 302  I = 5, 1, -1
        IF (PLANOK(I) .AND. DIST(I) .LE. 2100.0) GO TO 303
  302   CONTINUE
        IF (TRACE) PRINT 930,TIME,N
  930   FORMAT (F6.2,'   GUN',I4,' READY - NO TARGET')
        GUNRDY(N) = .TRUE.
        GO TO 99
  303   IF (TRACE) PRINT 931,TIME,N,I
  931   FORMAT (F6.2,'   GUN',I4,' FIRES AT AIRCRAFT',I4)
        CALL PUT(BURST, TIME+TOFFLT(I), I)
        CALL PUT(READY, TIME+NORMAL(5.0, 0.5, SEED), N)
      ENDIF
      GO TO 99
```

(*continued*)

```
C
C  THIS SECTION SIMULATES A SHELL BURST
C
  400 IF (PLANOK(N)) THEN
        IF (TRACE) PRINT 940,TIME,N
  940   FORMAT (F6.2,'   SHOT BURSTS BY AIRCRAFT',I4)
        IF (UNIF(SEED).GT.0.3-ABS(DIST(N))/10000.0) THEN
          IF (TRACE) PRINT 941,TIME,N
  941     FORMAT (F6.2,'   AIRCRAFT',I4,' DESTROYED')
          PLANOK(N) = .FALSE.
          PDOWN = PDOWN + 1
        ELSE
          IF (TRACE) PRINT 942,TIME
  942     FORMAT (F6.2,'   NO DAMAGE')
        ENDIF
      ELSE
        IF (TRACE) PRINT 943,TIME,N
  943   FORMAT (F6.2,'   SHOT AIMED AT AIRCRAFT',I4,
     X       ' BURSTS IN THE AIR')
      ENDIF
      GO TO 99
C
C  THIS IS THE END OF THE SIMULATION
C
  500 PRINT 950,GOUT,PDOWN
  950 FORMAT (///,I5,' GUNS DESTROYED',/,I5,
     X       ' PLANES SHOT DOWN')
      STOP
      END
```

Figure X.7.1. A simulation in Fortran.

```
      PROGRAM SIMNAI
      COMMON /TIMCOM/ TIME
      INTEGER ARRVAL,ENDSRV,OPNDOR,SHTDOR
      INTEGER ARRSD,SRVSD,TELSD,RNGSD
      INTEGER QUEUE,TLATWK,TLFREE,NCUST,NSERV
      INTEGER NREPLS,REPL,EVENT,I
      REAL    TIME,NXTARR,ERLANG,UNIF,X
      LOGICAL TRACE
      PARAMETER (ARRVAL = 1, ENDSRV = 2, OPNDOR = 3, SHTDOR = 4)
      EXTERNAL NXTARR, UNIF, ERLANG, INIT, PUT, GET, SETARR
C
C  EXAMPLE 1.4.1 - SEE SECTION 7.1.6
C
C  THE TIME UNIT IS 1 MINUTE.
C  TIME 0 CORRESPONDS TO 9 A.M.
C
C  CHANGE THE FOLLOWING TO SUPPRESS TRACE
C
      TRACE = .TRUE.
C
C  SET RANDOM NUMBER SEEDS SPACED 200000 VALUES APART
C
      ARRSD = 1234567890
      SRVSD = 1933985544
      TELSD = 2050954260
      RNGSD =  918807827
C
C  SET NUMBER OF RUNS TO CARRY OUT
C
      NREPLS = 200
      DO 98 REPL = 1,NREPLS
C
C  INITIALIZE FOR A NEW RUN
C
C  SET NUMBER OF TELLERS AT WORK.
C  INITIALLY NO TELLERS ARE FREE SINCE THE DOOR IS NOT OPEN
C
      X = UNIF(TELSD)
      TLATWK = 3
      IF (X.LT.0.20) TLATWK = 2
      IF (X.LT.0.05) TLATWK = 1
      TLFREE = 0
      IF (TRACE) PRINT 950,TLATWK
  950 FORMAT (I3,' TELLERS AT WORK')
C
C  SET QUEUE EMPTY, NO CUSTOMERS OR SERVICES SO FAR
C
      QUEUE = 0
      NCUST = 0
      NSERV = 0
C
C  SET TIME 0, SCHEDULE EMPTY, ARRIVAL DISTBN INITIALIZED
C
      CALL INIT
      X = SETARR(0)
C
C  INITIALIZE SCHEDULE WITH FIRST ARRIVAL, OPENING OF DOOR,
C  AND CLOSING OF DOOR ( = END OF 1 RUN)
C
      CALL PUT (ARRVAL, NXTARR(ARRSD))
      CALL PUT (OPNDOR, 60.0)
      CALL PUT (SHTDOR, 360.0)
```

(*continued*)

```
C
C  ******************
C
C  START OF MAIN LOOP
C
C  ******************
C
   99 CALL GET (EVENT)
      IF (EVENT .EQ. ARRVAL) GO TO 100
      IF (EVENT .EQ. ENDSRV) GO TO 200
      IF (EVENT .EQ. OPNDOR) GO TO 300
      IF (EVENT .EQ. SHTDOR) GO TO 400
C
C  THIS SECTION SIMULATES THE ARRIVAL OF A CUSTOMER
C
  100 NCUST = NCUST + 1
      IF (TRACE) PRINT 951,TIME,NCUST
  951 FORMAT (F10.2,' ARRIVAL OF CUSTOMER',I6)
C
C  AT EACH ARRIVAL, WE MUST SCHEDULE THE NEXT
C
      CALL PUT (ARRVAL, NXTARR(ARRSD))
C
C  IF THERE IS A TELLER FREE, WE START SERVICE
C
      IF (TLFREE.GT.0) THEN
        TLFREE = TLFREE - 1
        NSERV = NSERV + 1
        IF (TRACE) PRINT 952,TIME,NSERV
  952   FORMAT (F10.2,' START OF SERVICE',I6)
        CALL PUT (ENDSRV, TIME + ERLANG(2.0,2,SRVSD))
C
C  OTHERWISE THE CUSTOMER MAY OR MAY NOT STAY,
C  DEPENDING ON THE QUEUE LENGTH
C
      ELSE
        IF (UNIF(RNGSD).GT.(QUEUE-5)/5.0) THEN
          QUEUE = QUEUE + 1
          IF (TRACE) PRINT 953,TIME,QUEUE
  953     FORMAT (F10.2,' QUEUE IS NOW',I6)
        ELSE
          IF (TRACE) PRINT 954,TIME
  954     FORMAT (F10.2,' CUSTOMER DOES NOT JOIN QUEUE')
        ENDIF
      ENDIF
      GO TO 99
C
C  IN THIS SECTION WE SIMULATE THE END OF A PERIOD OF SERVICE
C
C  IF THE QUEUE IS NOT EMPTY, THE FIRST CUSTOMER WILL START
C  SERVICE AND WE MUST SCHEDULE THE CORRESPONDING END
C  SERVICE. OTHERWISE THE TELLER IS NOW FREE.
C
  200 IF (QUEUE.NE.0) THEN
        NSERV = NSERV + 1
        QUEUE = QUEUE - 1
        IF (TRACE) PRINT 955,TIME,NSERV
  955   FORMAT (F10.2,' START OF SERVICE',I6)
        IF (TRACE) PRINT 956,TIME,QUEUE
  956   FORMAT (F10.2,' QUEUE IS NOW',I6)
        CALL PUT (ENDSRV, TIME + ERLANG(2.0,2,SRVSD))
```

(*continued*)

```
      ELSE
        TLFREE = TLFREE + 1
        IF (TRACE) PRINT 957,TIME,TLFREE
  957   FORMAT (F10.2,' NUMBER OF TELLERS FREE',I6)
      ENDIF
      GO TO 99
C
C   IN THIS SECTION WE SIMULATE THE OPENING OF THE DOOR
C
C   ALL THE TELLERS BECOME AVAILABLE. IF THERE ARE CUSTOMERS
C   IN THE QUEUE, SOME OF THEM CAN START SERVICE AND WE MUST
C   SCHEDULE THE CORRESPONDING END SERVICE EVENTS.
C
  300 TLFREE = TLATWK
      IF (TRACE) PRINT 958,TIME
  958 FORMAT (F10.2,' THE DOOR OPENS')
      DO 301 I = 1,TLATWK
      IF (QUEUE.NE.0) THEN
        NSERV = NSERV + 1
        QUEUE = QUEUE -1
        TLFREE = TLFREE - 1
        IF (TRACE) PRINT 959,TIME,NSERV
  959   FORMAT (F10.2,' START OF SERVICE',I6)
        IF (TRACE) PRINT 960,TIME,QUEUE
  960   FORMAT (F10.2,' NUMBER IN QUEUE IS NOW',I6)
        CALL PUT (ENDSRV, TIME + ERLANG(2.0,2,SRVSD))
      ENDIF
  301 CONTINUE
      IF (TRACE) PRINT 961,TIME,TLFREE
  961 FORMAT (F10.2,' NUMBER OF TELLERS FREE',I6)
      GO TO 99
C
C   IN THIS SECTION WE SIMULATE THE CLOSING OF THE DOOR
C
C   SINCE WE KNOW ALL CUSTOMERS ALREADY IN THE QUEUE WILL
C   EVENTUALLY RECEIVE SERVICE, WE CAN SIMPLY COUNT THEM
C   AND STOP SIMULATING AT THIS POINT.
C
  400 CONTINUE
      IF (TRACE) PRINT 962,TIME,QUEUE
  962 FORMAT (F10.2,' DOOR CLOSES : NUMBER IN QUEUE',I6)
      NSERV = NSERV + QUEUE
      PRINT 990, TLATWK, NCUST, NSERV
  990 FORMAT (I6,' TELLERS,',I6,' CUSTOMERS,',I6,' WERE SERVED')
      WRITE (1,991) TLATWK,NCUST,NSERV
  991 FORMAT (3I6)
C
C   THIS IS THE END OF ONE RUN
C
   98 CONTINUE
      STOP
      END
```

Figure X.7.2. The standard example in Fortran.

```
      REAL FUNCTION ERLANG(MEAN,SHAPE,SEED)
      REAL MEAN,UNIF,X
      INTEGER SHAPE,SEED,I
      EXTERNAL UNIF
      X = 1.0
      DO 1 I = 1,SHAPE
    1 X = X * UNIF(SEED)
      ERLANG = -MEAN * ALOG(X) / SHAPE
      RETURN
      END
```

Figure X.7.3. The function ERLANG.

```
      REAL FUNCTION NXTARR(SEED)
      REAL X(4),Y(4),MEAN(4),CUMUL,UNIF
      INTEGER SEG,SEED
      EXTERNAL UNIF
      SAVE CUMUL, SEG, X, Y, MEAN
C
C   THIS SUBROUTINE IMPLEMENTS A PROCEDURE
C   FOR NONSTATIONARY PROCESSES: SEE 5.3.18, 4.9.
C   CUMUL GIVES THE SUCCESSIVE VALUES OF Y, WHILE SEG TELLS US
C   WHICH SEGMENT OF THE PIECEWISE-LINEAR INTEGRATED RATE
C   FUNCTION WE ARE CURRENTLY ON.
C
C   THE TABLE BELOW DEFINES THE PIECEWISE-LINEAR FUNCTION.
C   THE DUMMY VALUE OF Y(4) SERVES TO STOP US OVER-RUNNING
C   THE TABLE.
C
      DATA X    / 45.0,120.0,300.0,  0.0/
      DATA Y    /  0.0, 37.5,217.5, 1E10/
      DATA MEAN /  2.0,  1.0,  2.0,  0.0/
      CUMUL = CUMUL - ALOG(UNIF(SEED))
C
C   FIRST WE MUST CHECK IF WE ARE STILL IN THE SAME SEGMENT
C
    1 IF (CUMUL.LT.Y(SEG+1)) GO TO 2
      SEG = SEG + 1
      GO TO 1
C
C   NOW WE KNOW THE SEGMENT, WE CAN CALCULATE THE INVERSE
C
    2 NXTARR = X(SEG) + (CUMUL - Y(SEG)) * MEAN(SEG)
      RETURN
C
C   THIS ENTRY POINT INITIALIZES CUMUL AND SEG
C
      ENTRY SETARR
      CUMUL = 0.0
      SEG = 1
      SETARR = 0.0
      RETURN
      END
```

Figure X.7.4. The function NXTARR.

```
''EXAMPLE OF SIMSCRIPT MODELING

''EXAMPLE 7.2.1 OF SECTION 7.2.3

PREAMBLE

   EVENT NOTICES INCLUDE CUSTOMER.ARRIVAL, NEW.HAMBURGER,
      AND STOP.SIMULATION
      EVERY END.SERVICE HAS A PATRON

   TEMPORARY ENTITIES
      EVERY ORDER HAS A SIZE
         AND A TIME.OF.ARRIVAL
         AND MAY BELONG TO THE QUEUE
      DEFINE TIME.OF.ARRIVAL AS A REAL VARIABLE
      DEFINE SIZE AS AN INTEGER VARIABLE

   THE SYSTEM HAS A SERVER.STATUS
      AND A TOTAL.CUSTOMERS
      AND A LOST.CUSTOMERS
      AND A STOCK
      AND A SERVICE.TIME
      AND A STREAM.NO
      AND AN ORDER.SIZE RANDOM STEP VARIABLE
      AND OWNS THE QUEUE

   DEFINE SERVICE.TIME AS A REAL VARIABLE
   DEFINE TOTAL.CUSTOMERS, LOST.CUSTOMERS, STOCK,
      SERVER.STATUS AND STREAM.NO AS INTEGER VARIABLES
      DEFINE ORDER.SIZE AS AN INTEGER,STREAM 7 VARIABLE
      DEFINE QUEUE AS A FIFO SET

   ACCUMULATE UTILIZATION AS THE MEAN OF SERVER.STATUS
   TALLY MEAN.SERVICE.TIME AS THE MEAN OF SERVICE.TIME

   DEFINE .BUSY TO MEAN 1
   DEFINE .IDLE TO MEAN 0
   DEFINE SECONDS TO MEAN UNITS
   DEFINE MINUTES TO MEAN *60 UNITS
   DEFINE HOURS TO MEAN *60 MINUTES
END ''PREAMBLE


MAIN
   LET STREAM.NO = 1
   READ ORDER.SIZE
   SCHEDULE A STOP.SIMULATION IN 4 HOURS
   SCHEDULE A NEW.HAMBURGER IN
      UNIFORM.F(30.0,50.0,STREAM.NO) SECONDS
   SCHEDULE A CUSTOMER.ARRIVAL IN
      EXPONENTIAL.F(2.5,STREAM.NO) MINUTES
   START SIMULATION
END ''MAIN


EVENT CUSTOMER.ARRIVAL
   ADD 1 TO TOTAL.CUSTOMERS
   SCHEDULE A CUSTOMER.ARRIVAL IN
      EXPONENTIAL.F(2.5,STREAM.NO) MINUTES
   IF N.QUEUE = 3,
      ADD 1 TO LOST.CUSTOMERS
```

(*continued*)

```
      ELSE
         CREATE AN ORDER
         LET SIZE(ORDER) = ORDER.SIZE
         LET TIME.OF.ARRIVAL = TIME.V
         FILE ORDER IN QUEUE
         CALL START.SERVICE
      ALWAYS
      RETURN
   END ''EVENT CUSTOMER.ARRIVAL


   ROUTINE TO START.SERVICE
      IF SERVER.STATUS = .IDLE
         AND QUEUE IS NOT EMPTY
         AND SIZE(F.QUEUE) < STOCK
         REMOVE FIRST ORDER FROM QUEUE
         SUBTRACT SIZE(ORDER) FROM STOCK
         LET SERVER.STATUS = .BUSY
         SCHEDULE AN END.SERVICE GIVING ORDER IN
            45.0 + 25.0*SIZE(ORDER) SECONDS
      ALWAYS
      RETURN
   END ''ROUTINE TO START.SERVICE


   EVENT NEW.HAMBURGER
      ADD 1 TO STOCK
         SCHEDULE A NEW.HAMBURGER IN
         UNIFORM.F(30.0,50.0,STREAM.NO) SECONDS
      CALL START.SERVICE
      RETURN
   END ''EVENT NEW.HAMBURGER


   EVENT END.SERVICE GIVEN ORDER
      DEFINE ORDER AS AN INTEGER VARIABLE
      LET SERVICE.TIME = TIME.V - TIME.OF.ARRIVAL(ORDER)
      DESTROY THIS ORDER
      LET SERVER.STATUS = .IDLE
      CALL START.SERVICE
      RETURN
   END ''EVENT END.SERVICE


   EVENT STOP.SIMULATION
      PRINT 3 LINES WITH TOTAL.CUSTOMERS, LOST.CUSTOMERS,
         MEAN.SERVICE.TIME AND UTILIZATION THUS
*** CUSTOMERS ARRIVED.    ***WERE LOST.
MEAN SERVICE TIME WAS    *** SECONDS.
THE SERVER UTILIZATION WAS *.**
      STOP
   END ''STOP.SIMULATION
```

Figure X.7.5. A simulation in Simscript.

```
'' THE STANDARD EXAMPLE IN SIMSCRIPT II.5

'' THE TIME UNIT IS 1 MINUTE, WITH TIME 0 AT 9.00 A.M.

PREAMBLE

   EVENT NOTICES INCLUDE ARRIVAL, DEPARTURE, OPENING,
                         AND CLOSING

   TEMPORARY ENTITIES
      EVERY CUSTOMER HAS A TIME.OF.ARRIVAL AND MAY BELONG
                         TO THE QUEUE

   THE SYSTEM OWNS THE QUEUE

   DEFINE TEL.FREE, TEL.AT.WORK, NUM.CUSTOMERS,
          NUM.SERVICES, AND NUM.BALKS AS INTEGER VARIABLES
   DEFINE WAITING.TIME AS A REAL VARIABLE
   DEFINE NEXT.ARRIVAL AS A REAL FUNCTION

   DEFINE ARRIVAL.SEED TO MEAN 1
   DEFINE SERVICE.SEED TO MEAN 2
   DEFINE TELLER.SEED  TO MEAN 3
   DEFINE BALK.SEED    TO MEAN 4

   ACCUMULATE AVG.QUEUE AS THE AVERAGE OF N.QUEUE
   TALLY AVG.WAIT  AS THE AVERAGE OF WAITING.TIME

END

MAIN

   DEFINE X AS A REAL VARIABLE

   LET X = UNIFORM.F(0.0, 1.0, TELLER.SEED)
   LET TEL.AT.WORK = 3   '' NOW FIX UP NUMBER OF TELLERS
   IF X < 0.2 LET TEL.AT.WORK = 2
   ALWAYS IF X < 0.05 LET TEL.AT.WORK = 1

   ALWAYS

   LET TEL.FREE = 0
   LET NUM.CUSTOMERS = 0
   LET NUM.SERVICES  = 0
   LET NUM.BALKS     = 0

   SCHEDULE AN OPENING AT 60.0
   SCHEDULE A CLOSING  AT 360.0
   SCHEDULE AN ARRIVAL AT NEXT.ARRIVAL
   START SIMULATION

   ''CONTROL WILL RETURN WHEN ALL CUSTOMERS HAVE BEEN SERVED

   PRINT 3 LINES WITH TEL.AT.WORK,
      NUM.CUSTOMERS, NUM.SERVICES,
       NUM.BALKS, AVG.QUEUE, AND AVG.WAIT THUS
** TELLERS AT WORK
*** CUSTOMERS, *** SERVICES, *** BALKS
AVERAGE QUEUE *.**, AVERAGE WAIT *.** MINUTES

END
```

(*continued*)

```
EVENT ARRIVAL
   ADD 1 TO NUM.CUSTOMERS
   SCHEDULE AN ARRIVAL AT NEXT.ARRIVAL
   IF (N.QUEUE - 5)/5.0 > UNIFORM.F(0.0, 1.0, BALK.SEED)
      ADD 1 TO NUM.BALKS
      RETURN
   ALWAYS
   IF TEL.FREE = 0
      CREATE A CUSTOMER
      LET TIME.OF.ARRIVAL(CUSTOMER) = TIME.V
      FILE CUSTOMER IN QUEUE
   ELSE
      LET WAITING.TIME = 0.0
      SCHEDULE A DEPARTURE IN ERLANG.F(2.0, 2, SERVICE.SEED)
                                          UNITS
      ADD 1 TO NUM.SERVICES
      SUBTRACT 1 FROM TEL.FREE
   ALWAYS
   RETURN
END

EVENT DEPARTURE
   IF QUEUE IS EMPTY
      ADD 1 TO TEL.FREE
   ELSE
      REMOVE FIRST CUSTOMER FROM QUEUE
      LET WAITING.TIME = TIME.V - TIME.OF.ARRIVAL(CUSTOMER)
      DESTROY CUSTOMER
      SCHEDULE A DEPARTURE IN ERLANG.F(2.0, 2, SERVICE.SEED)
                                          UNITS
      ADD 1 TO NUM.SERVICES
   ALWAYS
   RETURN
END

EVENT OPENING
   DEFINE I AS AN INTEGER VARIABLE
   RESET TOTALS OF N.QUEUE
   FOR I = 1 TO TEL.AT.WORK DO
      SCHEDULE A DEPARTURE NOW
   LOOP
   '' FREEING ALL THE TELLERS HAS EXACTLY THE SAME EFFECT
   '' AS TEL.AT.WORK DEPARTURES
END

EVENT CLOSING
   CANCEL THE ARRIVAL
   RETURN
END

ROUTINE NEXT.ARRIVAL

   DEFINE X, Y, AND M AS REAL, SAVED, 1-DIMENSIONAL ARRAYS
   DEFINE CUMUL AS A REAL, SAVED VARIABLE
   DEFINE SEG AS AN INTEGER, SAVED VARIABLE

   IF SEG = 0 ''THIS IS THE FIRST CALL OF THE ROUTINE
      LET SEG = 1
      LET CUMUL = 0.0
      RESERVE X(*), Y(*), AND M(*) AS 4
      LET X(1) = 45.0
      LET X(2) = 120.0
```

(*continued*)

```
        LET X(3) = 300.0
        LET X(4) = 0.0
        LET Y(1) = 0.0
        LET Y(2) = 37.5
        LET Y(3) = 217.5
        LET Y(4) = 99999.9
        LET M(1) = 2.0
        LET M(2) = 1.0
        LET M(3) = 2.0
        LET M(4) = 0.0
    ALWAYS
    ADD EXPONENTIAL.F(1.0, ARRIVAL.SEED) TO CUMUL
    'SEARCH' IF CUMUL >= Y(SEG+1)
        ADD 1 TO SEG
        GO TO SEARCH
    ALWAYS
    RETURN WITH X(SEG) + (CUMUL - Y(SEG)) * M(SEG)

END
```

Figure X.7.6. The standard example in Simscript.

```
        SIMULATE

* EXAMPLE 7.3.1 OF SECTION 7.3.4
* 1 SIMULATED TIME UNIT REPRESENTS 100 MSEC.

EXPO    FUNCTION        RN8,C24
0.0,0/0.1,0.104/0.2,0.222/0.3,0.355/0.4,0.509/0.5,0.69
0.6,0.915/0.7,1.2/0.75,1.38/0.8,1.6/0.84,1.83/0.88,2.12
0.9,2.3/0.92,2.52/0.94,2.81/0.95,2.99/0.96,3.2/0.97,3.5
0.98,3.9/0.99,4.6/0.995,5.3/0.998,6.2/0.999,7.0/0.9997,8.0

CLASS   FUNCTION        RN7,D3
0.6,1/0.9,2/1.0,3

TIME    FUNCTION        PF1,L3
1,1/2,5/3,20

CORE    FUNCTION        PF1,L3
1,16/2,32/3,64

MEMRY   STORAGE         256
ALL     EQU             4,Q
CLS1    QTABLE          1,10,10,15
CLS2    QTABLE          2,10,10,15
CLS3    QTABLE          3,50,50,10
TOTAL   QTABLE          ALL,20,20,30

        GENERATE        600                 TIMER - DECREMENTS
        TERMINATE       1                   COUNTER EVERY MINUTE

        GENERATE        ,,,20,,3PF          CREATE 20 TRANSACTIONS
                                            WITH 3 FULL-WORD
                                            PARAMETERS EACH.
CYCLE   ADVANCE         100,FN$EXPO         TYPE A COMMAND.
        ASSIGN          1,FN$CLASS,PF
        ASSIGN          2,FN$TIME,PF
        ASSIGN          3,FN$CORE,PF
        QUEUE           PF1                 COMMAND TRANSMITTED.
        QUEUE           ALL
        ENTER           MEMRY,PF3           CONTEND FOR CORE.
RROB    SEIZE           CPU
        ADVANCE         1                   USE 1 QUANTUM OF CPU.
        RELEASE         CPU
        ASSIGN          2-,1,PF             IF REQUIRED TIME IS NOW
        TEST G          PF2,0,DONE          ZERO, GO TO DONE;
        BUFFER                              OTHERWISE GO TO BACK OF
        TRANSFER        ,RROB               CPU QUEUE AND WAIT TURN.
DONE    LEAVE           MEMRY,PF3           RELEASE CORE.
        DEPART          PF1                 REPLY READY : MEASURE
        DEPART          ALL                 RESPONSE TIME.
        TRANSFER        ,CYCLE

        START           1,NP                INITIALIZE WITHOUT
                                            PRINTING.
        RESET                               REINITIALIZE STATISTICS.
        START           10                  SIMULATE 10 MINUTES.
        END
```

Figure X.7.7. A simulation in GPSS.

```
       SIMULATE
*
* THE RUNNING EXAMPLE - SEE SECTION 1.4
*
* THE TIME UNIT IS 1 SECOND, WITH TIME 0 AT 9.45 A.M.
*     THUS TIME   900 = 10.00 A.M.
*                4500 = 11.00 A.M.
*               15300 =  2.00 P.M.
*               18900 =  3.00 P.M.
*
* THE FOLLOWING ENTITIES ARE USED :
*
* FACILITIES - TELL1, TELL2, TELL3 : THE THREE TELLERS
* QUEUES     - BANK  : KEEPS TRACK OF HOW MANY PEOPLE ARE
*                      IN THE SYSTEM
*            - WLINE : KEEPS TRACK OF HOW MANY PEOPLE ARE
*                      IN THE WAITING LINE
* SWITCHES   - DOOR  : SET WHEN THE DOOR OF THE BANK IS OPEN
*
* NOW DEFINE SOME NECESSARY FUNCTIONS :
*
* THE EXPONENTIAL DISTRIBUTION WITH MEAN 1
* THIS IS THE STANDARD APPROXIMATION TAKEN FROM THE
* IBM MANUAL
*
EXPON  FUNCTION  RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.83/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5
.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9997,8
*
* MEAN INTERARRIVAL TIME OF CUSTOMERS AS A FUNCTION
* OF THE TIME OF DAY
*
MEAN   FUNCTION  C1,D3
4500,120/15300,60/18900,120
*
* NUMBER OF TELLERS AT WORK
*
TELLS  FUNCTION  RN1,D3
.8,3/.95,2/1,1
*
* PROBABILITY OF BALKING ( TIMES 1000 ) AS A FUNCTION
* OF WAITING-LINE LENGTH
*
BALK   FUNCTION  Q$WLINE,D6
5,0/6,200/7,400/8,600/9,800/10,1000
*
* THE ERLANG DISTRIBUTION WITH MEAN AND SHAPE 2 :
* THIS IS CALCULATED SIMPLY AS THE SUM OF TWO EXPONENTIALS
*
ERLAN  FVARIABLE  60*(FN$EXPON+FN$EXPON)
*
* THE FOLLOWING SECTION GIVES THE CODE FOR A CUSTOMER.
* AS DISCUSSED IN THE TEXT, THE ARRIVAL DISTRIBUTION IS
* FUDGED AT THE POINTS WHERE THE ARRIVAL RATE CHANGES.
* A 'CUSTOMER' TRANSACTION HAS ONE HALF-WORD PARAMETER
* USED TO HOLD THE NUMBER OF THE TELLER HE IS USING.
*
```

(*continued*)

```
        GENERATE    FN$MEAN,FN$EXPON,,,,1PH   GENERATE A NEW
*                                             CUSTOMER
        ADVANCE     0           TO CLEAR THE 'GENERATE' BLOCK
        GATE LS     DOOR        IF THE DOOR IS CLOSED, WAIT
        QUEUE       BANK        ENTER THE SYSTEM
        TRANSFER    .FN$BALK,,BALK   BALK IF QUEUE TOO LONG
        QUEUE       WLINE       ENTER WAITING LINE
        TRANSFER    ALL,GET1,GET3,3   TRY ALL THREE TELLERS
GET1    SEIZE       TELL1
        ASSIGN      1,1,PH      GET TELLER 1
        TRANSFER    ,SERV
GET2    SEIZE       TELL2
        ASSIGN      1,2,PH      GET TELLER 2
        TRANSFER    ,SERV
GET3    SEIZE       TELL3
        ASSIGN      1,3,PH      GET TELLER 3
SERV    DEPART      WLINE       LEAVE WAITING LINE
        ADVANCE     V$ERLAN     RECEIVE SERVICE
        RELEASE     PH1         TELLER IS FREED
        DEPART      BANK        LEAVE THE SYSTEM
        TERMINATE
*
BALK    DEPART      BANK        AFTER BALKING, LEAVE THE SYSTEM
        TERMINATE
*
* THIS SECTION HANDLES THE SETUP OF TELLERS, OPENS
* AND CLOSES THE DOOR OF THE BANK, AND CONTROLS
* THE ENDING OF THE SIMULATION
*
        GENERATE    1,,,1,127,1PH   GENERATE ONE TOP-PRIORITY
*                                   TRANSACTION
        FUNAVAIL    TELL1-TELL3     AT 9.45 ALL TELLERS ARE
*                                   UNAVAILABLE
        ADVANCE     899         WAIT UNTIL 10.00
        LOGIC S     DOOR        OPEN THE DOOR
        ASSIGN      1,FN$TELLS,PH
        FAVAIL      TELL1
        TEST G      PH1,1,*+4   MAKE 1 TO 3 TELLERS
        FAVAIL      TELL2       AVAILABLE
        TEST G      PH1,2,*+2
        FAVAIL      TELL3
        ADVANCE     18000       WAIT UNTIL 3.00
        LOGIC R     DOOR        CLOSE THE DOOR
        TEST E      Q$BANK,0    WAIT UNTIL CUSTOMERS LEAVE
        TERMINATE   1           END RUN
*
*** CONTROL CARDS ***
*
        START       1
        END
```

Figure X.7.8. The standard example in GPSS.

```
simulation begin

comment Example 7.4.1 of section 7.4.3;

process class passenger;
     begin
     Boolean taxpaid, repeat;
     real timein;
     taxpaid := draw (0.75, seeda);
     timein := time;
     activate new passenger delay  negexp (1, seedb);
     if not freeclerks.empty then begin
                 activate freeclerks.first after current;
                 freeclerks.first.out end;
     wait (checkinq)
     end passenger;



process class clerk;
     begin
clerkloop: inspect checkinq.first
     when passenger do begin
           checkinq.first.out;
           if repeat then hold (1) else hold (normal
                 (3,1,seedc));
           if taxpaid then begin
                 totpass := totpass + 1;
                 totwait := totwait + time - time in end
           else begin
                 if not freetaxmen.empty then begin
                       activate freetaxmen.first after current;
                       freetaxmen.first.out end;
                 this passenger.into (taxq) end end
        otherwise wait (free clerks);
        goto clerkloop
        end clerk;



process class taxman;
     begin
taxloop: inspect taxq.first
     when passenger do begin
           taxq.first.out; hold (negexp (2,seedd));
           taxpaid := repeat := true;
           if not freeclerks.empty then begin
                 activate freeclerks.first after current;
                 freeclerks.first.out end;
           this passenger.into (checkinq) end
     otherwise wait (freetaxmen);
     goto taxloop
     end taxman;
     integer seeda, seedb, seedc, seedd, i;
     integer nclerks, ntaxmen, totpass;
     real totwait;
     ref (head) checkinq, taxq, freeclerks, freetaxmen;
```

(continued)

```
seeda := 12345;   {and so on for the other seeds}
nclerks := inint; ntaxmen := inint;   {read these values}
checkinq :- new head; freetaxmen :- new head;
for i := 1  step 1 until nclerks do activate new clerk;
for i := 1  step 1 until ntaxmen do activate new taxman;
activate new passenger delay negexp (1, seedb);
hold (120);
outtext ('number of clerks'); outint (nclerks, 6); outimage;
outtext ('number of taxmen'); outint (ntaxmen, 6); outimage;
outtext ('mean wait'); outfix (totwait/totpass, 2,6); outimage

end simulation
```

Figure X.7.9. A simulation in Simula.

```
begin external class Demos;

comment Example 7.3.1 of section 7.3.4 recoded in Demos;

Demos begin
      ref (negexp) thinktime;
      ref (randint) aux ;
      ref (res) core, cpu ;
      ref (histogram) overall ;
      ref (histogram) array response [1:3] ;
      integer array time needed, core needed [1:3] ;
      integer  i ;

      comment The unit of time is 1 msec ;

      entity class terminal ;
             begin
             integer class, time left, i ;
             real time began ;
             hold (thinktime.sample) ;
             i := aux.sample ;
             class := if i < 60 then 1 else
                          if i < 90 then 2 else 3 ;
             time began := time ;
             time left := time needed [class] ;
             core.acquire (core needed [class]) ;
             while time left > 0 do
                   begin cpu.acquire (1) ;
                         hold (100) ;
                         time left := time left -100 ;
                         cpu.release (1)
                   end ;
             core.release (core needed [class]) ;
             overall.update (time-time began) ;
             response [class].update (time-time began) ;
             repeat
             end terminal ;
             thinktime :- new negexp ("THINK TIME", 0.0001) ;
             aux    :- new randint ("AUXILIARY", 0,99) ;
             core   :- new res ("CORE", 256) ;
             cpu    :- new res ("CPU", 1) ;
             overall :- new histogram ("OVERALL", 0.0, 60000.0, 30) ;
             response [1] :-
                new histogram ("CLASS 1", 0.0, 15000.0, 15) ;
             response [2] :-
                new histogram ("CLASS 2", 0.0, 15000.0, 15) ;
             response [3] :-
                new histogram ("CLASS 3", 0.0, 60000.0, 30) ;
             time needed [1] := 100 ;
             time needed [2] := 500 ;
             time needed [3] := 2000 ;
             core needed [1] := 16 ;
             core needed [2] := 32 ;
             core needed [3] := 64 ;

             for  i := 1 step 1 until 20 do
                    new terminal ("TERMINAL").schedule (0.0) ;

             hold (60000.0) ;
             reset ;
             hold (20.0 * 60000.0)
             end
      end
```

Figure X.7.10. A simulation using Demos.

```
begin
external class Demos;
Demos begin

comment   -   the running example - see section 1.4.

The time unit is one minute with time 0 at 9.00 a.m.;

integer n;
real x;
ref (count) arrivals, services, balks;
ref (erlang) servicetime;
ref (res) tellers;
ref (arrdist) nextarrival;
ref (balkdist) balk;

rdist class arrdist;
begin
real cumul; integer seg;
array x, y, mean [1 : 4];
ref (negexp) aux;

   real procedure sample;
   begin
   cumul := cumul + aux . sample;
   while cumul >= y[seg+1] do seg := seg + 1;
   sample := x[seg] + (cumul - y[seg]) * mean[seg] - time
   end;

x[1] :=  45.0; y[1] :=   0.0; mean[1] := 2.0;
x[2] := 120.0; y[2] :=  37.5; mean[2] := 1.0;
x[3] := 300.0; y[3] := 217.5; mean[3] := 2.0;
x[4] :=   0.0; y[4] := 1.0&9; mean[4] := 0.0;

aux :- new negexp ("arr aux", 1.0);
cumul := 0.0;
seg := 1
end *** definition of arrdist *** ;

bdist class balkdist (pr);
ref (res) pr;
begin

   Boolean procedure sample;
   sample := (pr.length - 5) / 5 > zyqsample;

   comment - zyqsample is the (0,1) uniform generator
             supplied by Demos;

end *** definition of balkdist *** ;
entity class customer;
begin
arrivals.update (1);
new customer ("customer") . schedule (nextarrival . sample);
if not balk . sample then
   begin
   tellers . acquire (1);
   services . update (1);
   hold (servicetime . sample);
   tellers . release (1)
   end
```

(continued)

```
else
   balks . update (1)
end *** definition of customer *** ;

trace;
   arrivals    :- new count ("arrivals");
   services    :- new count ("services");
   balks       :- new count ("balks");
   servicetime :- new erlang ("servicetime", 2.0, 2);
   nextarrival :- new arrdist ("interval");

   comment - draw number of tellers;
   x := new uniform ("aux unif", 0.0, 1.0) . sample;
   n := if x < 0.05 then 1
        else if x < 0.2 then 2
           else 3;

   comment - set up required tellers, though they are not
             yet available;
   tellers :- new res ("tellers", n);
   balk    :- new balkdist ("balks", tellers);
   tellers . acquire (n);

   comment - schedule first customer;
   new customer ("customer") . schedule (nextarrival . sample);

   comment - open door and make tellers available at 10.00 a.m.;
   hold (60.0);
   tellers . release (n);

   comment - wait until 3.00 p.m. and wrap up;
   hold (300.0);
   services . update (tellers . length)

end
end;
```

Figure X.7.11. The standard example in Demos.

```
      REAL FUNCTION ERLANG (MEAN, SHAPE, SEED)
      INTEGER I,J,K,SEED
      REAL X(25),FX(25),U,UNIF
      EXTERNAL UNIF
      DATA X /.0,.2,.4,.6,.8,1.0,1.2,1.4,1.6,1.8,2.0,
     X        2.2,2.4,2.6,2.8,3.0,3.2,3.4,3.6,3.8,4.0,
     X        5.0,6.0,7.0,7.79998/
      DATA FX/.0,.01752,.06155,.12190,.19121,
     X        .26424,.33737,.40817,.47507,.53716,
     X        .59399,.64543,.69156,.73262,.76892,
     X        .80085,.82880,.85316,.87431,.89262,
     X        .90842,.95957,.98265,.99270,1.0/
C
C   THIS FUNCTION GENERATES AN ERLANG WITH MEAN 2.0
C   AND SHAPE 2 USING LINEAR INTERPOLATION IN A TABLE
C   OF THE DISTRIBUTION
C
C   THE PARAMETERS MEAN AND SHAPE ARE NOT USED, BUT
C   REMAIN FOR COMPATIBILITY WITH THE MAIN PROGRAM.
C
      U = UNIF(SEED)
      I = 1
      J = 25
C
C   BINARY SEARCH FOR U IN TABLE
C
    1 K = (I + J)/2
      IF (FX(K).LE.U) GO TO 2
      J = K - 1
      GO TO 1
    2 IF (FX(K+1).GT.U) GO TO 3
      I = K + 1
      GO TO 1
    3 ERLANG = X(K) + (U - FX(K)) * (X(K+1) - X(K)) /
     X                (FX(K+1) - FX(K))
      RETURN
      END
```

Figure X.8.1. A table-driven Erlang generator.

```
C
C   INITIALIZE FOR A NEW RUN
C
       IF (EVEN) GO TO 96
C
C   ON AN ODD-NUMBERED RUN, WE START A NEW SEGMENT
C   OF RANDOM NUMBERS AND SET AVGSRV = 2.0
C
       DO 95 I = 1,100
       X = UNIF(ARRSD)
       X = UNIF(SRVSD)
   95  X = UNIF(RNGSD)
       HLDARR = ARRSD
       HLDSRV = SRVSD
       HLDRNG = RNGSD
       AVGSRV = 2.0
       GO TO 97
C
C   ON AN EVEN-NUMBERED RUN, WE USE THE SAME SEEDS
C   AS BEFORE FOR ALL GENERATORS, BUT SET AVGSRV = 1.8
C
   96  ARRSD = HLDARR
       SRVSD = HLDSRV
       RNGSD = HLDRNG
       AVGSERV = 1.8
C
   97  EVEN = .NOT.  EVEN
C
```

Figure X.8.2. Additional code when using common random numbers.

```
      PROGRAM STAREG
      INTEGER NTELL,NARR,NSERV,N,NM,I
      REAL X(200),Y(200),Z
      REAL ARR,SERV,SLOPE
      REAL SIGX,SIGY,SIGXY,SIGX2,SIGY2,SIGZ,SIGZ2
C
C  REGRESSION OF NSERV = Y AGAINST NARR = X
C  WE SUBTRACT OFF THE KNOWN MEAN 247.5 OF X
C  TO MAKE LIFE MORE SIMPLE
C
C  ESTIMATION IS BY SPLITTING
C
      N = 0
      SIGX = 0.0
      SIGY = 0.0
      SIGXY = 0.0
      SIGX2 = 0.0
      SIGY2 = 0.0
C
C  GET THE DATA AND CALCULATE THE APPROPRIATE SUMS
C
    1 READ (*,900, END = 2) NTELL,NARR,NSERV
  900 FORMAT (3I6)
C
C  NTELL IS IN FACT CONSTANT = 2
C
      ARR = NARR - 247.5
      SERV = NSERV
      N = N + 1
      SIGX = SIGX + ARR
      SIGY = SIGY + SERV
      SIGXY = SIGXY + ARR * SERV
      SIGX2 = SIGX2 + ARR * ARR
      SIGY2 = SIGY2 + SERV * SERV
      X(N) = ARR
      Y(N) = SERV
      GO TO 1
C
C  START BY PRINTING THE NAIVE ESTIMATE AS A CHECK
C
    2 PRINT 901,SIGY/N,(SIGY2 - SIGY*SIGY/N) /
     X     (N * (N-1))
  901 FORMAT (' NAIVE EST',F10.4,' EST VAR',F10.4)
C
C  NOW DO REGRESSION WITH SPLITTING
C  WE WILL CALL THE OBSERVATIONS Z
C
      NM = N - 1
      SIGZ = 0.0
      SIGZ2 = 0.0
      DO 3 I = 1,N
C
C  SUBTRACT OFF CURRENT X AND Y
C
      SIGX = SIGX - X(I)
      SIGY = SIGY - Y(I)
      SIGXY = SIGXY - X(I)*Y(I)
      SIGX2 = SIGX2 - X(I)*X(I)
      SIGY2 = SIGY2 - Y(I)*Y(I)
C
C  CALCULATE THE SLOPE USING THE REMAINING VALUES
```

(*continued*)

```
C
      SLOPE = (NM*SIGXY - SIGX*SIGY) / (NM*SIGX2 - SIGX*SIGX)
C
C  AND HENCE ONE OBSERVATION
C
      Z = Y(I) - SLOPE * X(I)
      SIGZ = SIGZ + Z
      SIGZ2 = SIGZ2 + Z * Z
C
C  PUT BACK THE CURRENT X AND Y
C
      SIGX = SIGX + X(I)
      SIGY = SIGY + Y(I)
      SIGXY = SIGXY + X(I)*Y(I)
      SIGX2 = SIGX2 + X(I)*X(I)
      SIGY2 = SIGY2 + Y(I)*Y(I)
    3 CONTINUE
C
C  WE SHOULD NOW HAVE A BETTER ESTIMATE
C
      PRINT 902,SIGZ/N,(SIGZ2 - SIGZ*SIGZ/N) /
     X      (N * (N-1))
  902 FORMAT (' EST WITH SPLITTING',F10.4,' EST VAR',
     X      F10.4,' --- BIASED --- ')
      STOP
      END
```

Figure X.8.3. Variance reduction by splitting.

```
      PROGRAM STASPL
      INTEGER NTELL,NARR,NSERV,N,NM,I
      REAL X(200),Y(200),ZN,Z
      REAL ARR,SERV,SLOPE
      REAL SIGX,SIGY,SIGXY,SIGX2,SIGY2,SIGZ,SIGZ2
C
C  REGRESSION OF NSERV = Y AGAINST NARR = X
C  WE SUBTRACT OFF THE KNOWN MEAN 247.5 OF X
C  TO MAKE LIFE MORE SIMPLE
C
C  ESTIMATION IS BY JACKKNIFING
C
      N = 0
      SIGX = 0.0
      SIGY = 0.0
      SIGXY = 0.0
      SIGX2 = 0.0
      SIGY2 = 0.0
C
C  GET THE DATA AND CALCULATE THE APPROPRIATE SUMS
C
    1 READ (*,900, END = 2) NTELL,NARR,NSERV
  900 FORMAT (3I6)
C
C     NTELL IS IN FACT CONSTANT = 2
C
      ARR = NARR - 247.5
      SERV = NSERV
      N = N + 1
      SIGX = SIGX + ARR
      SIGY = SIGY + SERV
      SIGXY = SIGXY + ARR * SERV
      SIGX2 = SIGX2 + ARR * ARR
      SIGY2 = SIGY2 + SERV * SERV
      X(N) = ARR
      Y(N) = SERV
      GO TO 1
C
C  START BY PRINTING THE NAIVE ESTIMATE AS A CHECK
C
    2 PRINT 901,SIGY/N,(SIGY2 - SIGY*SIGY/N) /
     X     (N * (N-1))
  901 FORMAT (' NAIVE EST',F10.4,' EST VAR',F10.4)
C
C  FIRST CALCULATE ZN USING ALL N VALUES
C
      ZN = (SIGY*SIGX2 - SIGX*SIGXY) / (N*SIGX2 - SIGX*SIGX)
      PRINT 903,ZN
  903 FORMAT (' ESTIMATE USING ALL N VALUES',F10.4)
C
C  NOW CALCULATE THE PSEUDOVALUES
C
      NM = N - 1
      SIGZ = 0.0
      SIGZ2 = 0.0
      DO 3 I = 1,N
C
C  SUBTRACT OFF CURRENT X AND Y
C
      SIGX = SIGX - X(I)
      SIGY = SIGY - Y(I)
```

(*continued*)

```
      SIGXY = SIGXY - X(I)*Y(I)
      SIGX2 = SIGX2 - X(I)*X(I)
      SIGY2 = SIGY2 - Y(I)*Y(I)
C
C  CALCULATE AN ESTIMATE USING ONLY N-1 VALUES
C
      Z = (SIGY*SIGX2 - SIGX*SIGXY) / (NM*SIGX2 - SIGX*SIGX)
C
C  AND HENCE CALCULATE THE NEXT PSEUDOVALUE
C
      Z = N * ZN - NM * Z
      SIGZ = SIGZ + Z
      SIGZ2 = SIGZ2 + Z * Z
C
C  PUT BACK THE CURRENT X AND Y
C
      SIGX = SIGX + X(I)
      SIGY = SIGY + Y(I)
      SIGXY = SIGXY + X(I)*Y(I)
      SIGX2 = SIGX2 + X(I)*X(I)
      SIGY2 = SIGY2 + Y(I)*Y(I)
    3 CONTINUE
C
C  WE SHOULD NOW HAVE A BETTER ESTIMATE
C
      PRINT 902,SIGZ/N,(SIGZ2 - SIGZ*SIGZ/N) /
     X     (N * (N-1))
  902 FORMAT (' EST WITH JACKKNIFING',F10.4,' EST VAR',
     X     F10.4,' --- BIASED --- ')
      STOP
      END
```

Figure X.8.4. Variance reduction by jackknifing.

```
            REAL FUNCTION B(SRVSD)
            REAL X,UNIF,ERLANG,BC,BT,BLIM,BMULT
            INTEGER SRVSD
            EXTERNAL UNIF, ERLANG
            COMMON /BPARAM/ BC,BT,BLIM,BMULT
C
            IF (UNIF(SRVSD).GT.BLIM) THEN
C
C  USE TRUNCATED ERLANG
C
     1      B = ERLANG(2.0,2,SRVSD)
            IF (B.GT.BT) GO TO 1
            BMULT = BMULT * BC
          ELSE
C
C  USE TRUNCATED EXPONENTIAL
C
            B = BT - ALOG(UNIF(SRVSD))
            BMULT = BMULT * BC * B / BT
          ENDIF
          RETURN
          END
```

Figure X.8.5. Importance sampling—the function B.

References

Abramovitz, M., and Stegun, I. A. (1964). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover: New York.

Adelson, R. M. (1966). Compound Poisson distributions. *Oper. Res. Quart.*, **17**, 73–75.

Ahrens, J. H., and Dieter, U. (1972a). Computer methods for sampling from the exponential and normal distributions. *Commun. ACM*, **15**, 873–882.

Ahrens, J. H., and Dieter, U. (1972b). Computer methods for sampling from gamma, beta, Poisson and binomial distributions. *Computing*, **12**, 223–246.

Ahrens, J. H., and Dieter, U. (1980). Sampling from binomial and Poisson distributions: a method with bounded computation times. *Computing*, **25**, 193–208.

Ahrens, J. H., and Dieter, U. (1982a). Generating gamma variates by a modified rejection technique. *Commun. ACM*, **25**, 47–54.

Ahrens, J. H., and Dieter, U. (1982b). Computer generation of Poisson deviates from modified normal distributions. *ACM Trans. Math. Software*, **8**, 163–179.

Ahrens, J. H., and Kohrt, K. D. (1981). Computer methods for efficient sampling from largely arbitrary statistical distributions. *Computing*, **26**, 19–31.

Albin, S. L. (1982). On the Poisson approximations for superposition arrival processes in queues. *Manage. Sci.*, **28**, 126–137.

Anderson, T. W., and Darling, D. A. (1954). A test of goodness of fit. *JASA*, **49**, 765–769.

Andrews, D. F., Bickel, P. J., Hampel, F. R., Huber, P. J., Rogers, W. H., and Tukey, J. W. (1972). *Robust Estimates of Location.* Princeton University Press: Princeton, N.J.

ANSI (1978). *American National Standard Programming Language FORTRAN.* American National Standards Institute: New York.

Arvidsen, N. I., and Johnsson, T. (1982). Variance reduction through negative correlation, a simulation study. *J. Stat. Comput. Simul.*, **15**, 119–127.

Avriel, M. (1976). *Nonlinear Programming: Analysis and Methods.* Prentice-Hall: Englewood Cliffs, N.J.

Babad, J. M. (1975). The IBM GPSS random number generator. Technical Report, University of Chicago, Graduate School of Business.

Barlow, R. E., and Proschan, F. (1965). *Mathematical Theory of Reliability.* Wiley: New York.

Barr, D. R., and Slezak, N. L. (1972). A comparison of multivariate normal generators. *Commun. ACM*, **15**, 1048–1049.

Bazaraa, M. S., and Shetty, C. M. (1979). *Nonlinear Programming: Theory and Algorithms*. Wiley: New York.

Bennett, C. H. (1979). On random and hard-to-describe numbers. IBM Watson Research Center Report RC 7483 (No. 32272), Yorktown Heights, N.Y.

Bentley, J. L., and Saxe, J. B. (1980). Generating sorted lists of random numbers. *ACM Trans. Math. Software*, **6**, 359–364.

Bergström, H. (1952). On some expansions of stable distributions. *Ark. Mat.* II, **18**, 375–378.

Berman, M. B. (1970). Generating random variates from gamma distributions with non-integer shape parameters. Report R-641-PR, The RAND Corporation, Santa Monica, Calif.

Best, D. J., and Roberts, D. E. (1975). Algorithm AS 91: the percentage points of the χ^2 distribution. *Appl. Stat.*, **24**, 385–388.

Bhattacharjee, G. P. (1970). Algorithm AS 32—The incomplete gamma integral. *Appl. Stat.*, **19**, 285–287.

Billingsley, P. (1968). *Convergence of Probability Measures*. Wiley, New York.

Birtwhistle, G. M. (1979a). *Demos Reference Manual*. Dep. of Computer Science, University of Bradford: Bradford.

Birtwhistle, G. M. (1979b). *Demos—Discrete Event Modelling on Simula*. Macmillan: London.

Birtwhistle, G. M., Dahl, O.-J., Myhrhaug, B., and Nygaard, K. (1973). *Simula Begin*. Studentlitteratur: Oslo.

Bobillier, P. A., Kahan, B. C., and Probst, A. R. (1976). *Simulation with GPSS and GPSS V*. Prentice-Hall: Englewood Cliffs, N.J.

Boender, C. G. E., Rinnooy Kan, A. H. G., Timmer, G. T., and Stougie, L. (1982). A stochastic method for global optimization. *Math. Program.*, **22**, 125–140.

Box, G. E. P., and Muller, M. E. (1958). A note on the generation of random normal deviates. *Ann. Math. Stat.*, **29**, 610–611.

Bright, H. S., and Enison, R. L. (1979). Quasi-random number sequences from a long-period TLP generator with remarks on application to cryptography. *Comput. Surv.*, **11**, 357–370.

Brillinger, D. R. (1981). *Time Series: Data Analysis and Theory*. Holden-Day: San Francisco.

Brinch Hansen, P. (1973). *Operating System Principles*. Prentice-Hall: Englewood Cliffs, N.J.

Brinch Hansen, P. (1977). *The Architecture of Concurrent Programs*. Prentice-Hall: Englewood Cliffs, N.J.

Brown, C. A., and Purdom, P. W. (1981). An average time analysis of backtracking. *SIAM J. Comput.*, **10**, 583–593.

Brown, M., and Solomon, H. (1979). On combining pseudorandom number generators. *Ann. Stat.*, **1**, 691–695.

Brown, M., Solomon, H., and Stephens, M. A. (1981). Monte Carlo simulation of the renewal process. *J. Appl. Probab.*, **18**, 426–434.

Brumelle, S. L. (1971). Some inequalities for parallel service queues. *Oper. Res.*, **19**, 402–413.

CACI Inc. (1976a). *Simscript II.5 Reference Handbook*. CACI Inc.: Los Angeles.

CACI Inc. (1976b). *Simulating with Processes and Resources in Simscript II.5*. CACI Inc.: Los Angeles.

Callaert, H., and Janssen, P. (1981). The convergence rate of fixed-width confidence intervals for the mean. *Sankyā*, **43**, 211–219.

Carson, J. S. (1978). Variance reduction techniques for simulated queuing processes. Report No. 78–8, Dept. of Industrial Engineering, University of Wisconsin, Madison.

Carson, J. S. (1980). Linear combinations of indirect estimators for variance reduction in regenerative simulations. Technical Report, Dept. of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta.

Carson, J. S., and Law, A. M. (1980). Conservation equations and variance reduction in queuing simulations. *Oper. Res.*, **28**, 535–546.

Carter, G., and Ignall, E. J. (1975). Virtual measures: a variance reduction technique for simulation. *Manage. Sci.*, **21**, 607–617.

CDC (1972). *MIMIC Digital Simulation Language Reference Manual.* Publication No. 44610400. Control Data Corporation: Sunnyvale, Calif.

Chambers, J. M., Mallows, C. L., and Stuck, B. W. (1976). A method for simulating stable random variables. *JASA*, **71**, 340–344.

Chen, H. C., and Asau, Y. (1974). On generating random variates from an empirical distribution. *AIEE Trans.*, **6**, 163–166.

Cheng, R. C. H. (1978). Generating beta variates with nonintegral shape parameters. *Commun. ACM*, **21**, 317–322.

Cheng, R. C. H. (1981). The use of antithetic control variates in computer simulation. *Proceedings 1981 Winter Simulation Conference*, pp. 313–318.

Cheng, R. C. H. (1982). The use of antithetic variates in computer simulations. *J. Opl. Res. Soc.*, **33**, 229–237.

Cheng, R. C. H., and Feast, G. M. (1979). Some simple gamma variate generators. *Appl. Stat.*, **28**, 290–295.

Chouinard, A., and McDonald, D. (1982). A rank test for non-homogeneous Poisson processes. Manuscript, Dept. of Mathematics, University of Ottawa.

Chow, Y. S., and Robbins, H. (1965). On the asymptotic theory of fixed-width sequential confidence intervals for the mean. *Ann. Math. Stat.*, **36**, 457–462.

Chu, Y. (1969). *Digital Simulation of Continuous Systems.* McGraw-Hill: New York.

Çinlar, E. (1972). Superposition of point processes. In *Stochastic Point Processes: Statistical Analysis, Theory, and Applications*, (P. A. W. Lewis, editor), pp. 549–606. Wiley: New York.

Çinlar, E. (1975). *Introduction to Stochastic Processes.* Prentice-Hall: Englewood Cliffs, N.J.

Clark, G. M. (1981). Use of Polya distributions in approximate solutions to non-stationary $M/M/s$ Queues. *Commun. ACM*, **24**, 206–217.

Clark, R. L. (1973). A linguistic contribution to GOTO-less programming. *Datamation*, December, 62–63.

Clementson, A. T. (1973). *ECSL User's Manual.* Department of Engineering Production, University of Birmingham: Birmingham.

Cochran, W. G. (1977). *Sampling Techniques.* Wiley: New York.

Conte, S. D., and de Boor, C. (1980). *Elementary Numerical Analysis.* McGraw-Hill: New York.

Conway, R. W. (1963). Some tactical problems in digital simulation. *Manage. Sci.*, **10**, 47–61.

Conway, R. W., Maxwell, W. L., and Miller, L. W. (1967). *Theory of Scheduling.* Addison-Wesley: Reading, Mass.

Cooper, R. B. (1981). *Introduction in Queueing Theory*, 2nd edn. North Holland: New York.

Coveyou, R. R., and McPherson, R. D. (1967). Fourier analysis of uniform random number generators. *J. ACM*, **14**, 100–119.

Cox, D. R., and Smith, W. L. (1961). *Queues.* Wiley: New York.

Crane, M. A., and Lemoine, A. J. (1977). *An Introduction to the Regenerative Method for Simulation Analysis.* (Lecture Notes in Control and Information Sciences.) Springer-Verlag: New York, Heidelberg, Berlin.

Dahl, O.-J., Myhrhaug, B., and Nygaard, K. (1970). *Common Base Language.* Publication S-22. Norwegian Computing Center: Oslo.

Dahl, O.-J., and Nygaard, K. (1967). *Simula: A Language for Programming and Description of Discrete Event Systems. Introduction and user's manual*, 5th edn. Norwegian Computing Center: Oslo.

Dannenbring, D. G. (1977). Procedures for estimating optimal solution values for large combinatorial problems. *Manage. Sci.*, **23**, 1273–1283.

Davey, D. (1982). Une contribution aux méthodes de construction des langages de simulation discrets. Ph.D. Thesis, Département d'informatique et de recherche opérationnelle, Université de Montréal.

Davey, D., and Vaucher, J. G. (1980). Self-optimizing partitioned sequencing sets for discrete event simulation. *INFOR*, **18**, 41–61.

Deák, I. (1980). Three digit accurate multiple normal probabilities. *Numer. Math.*, **35**, 369–380.

De Millo, R. A., Lipton, R. J., and Perlis, A. J. (1979). Social processes and proofs of theorems and programs. *Commun. ACM*, **22**, 271–280.

Denardo, E. V., and Fox, B. L. (1979). Shortest-route methods: 1, reaching, pruning, and buckets. *Oper. Res.* **27**, 161–186.

Devroye, L. (1979). Inequalities for the completion times of stochastic PERT networks. *Math. Oper. Res.*, **4**, 441–447.

Devroye, L. (1980). The computer generation of binomial random variables. Technical Report, School of Computer Science, McGill University, Montreal.

Devroye, L. (1981a). The computer generation of Poisson random variables. *Computing*, **26**, 197–207.

Devroye, L. (1981b). Random variate generation for unimodal and monotone densities. Technical Report, School of Computer Science, McGill University, Montreal. To appear in *Computing*.

Devroye, L. (1981c). Personal communication and unpublished course notes.

Devroye, L. (1982a). On the generalization of a sample for random variate generation. Manuscript, McGill University, Montreal.

Devroye, L. (1982b). A simple algorithm for generating random variates with a log-concave density. Manuscript, McGill University, Montreal.

Devroye, L. (1982c). A note on approximations in random variate generation. *J. Stat. Comput. Simul.*, **14**, 149–158.

Dieter, U. (1971). Pseudo-random numbers: the exact distribution of pairs. *Math. Comput.*, **25**, 855–883.

Dieter, U. (1975). How to calculate shortest vectors in a lattice. *Math. Comput.*, **29**, 827–833.

Dieter, U. (1982). Personal communication.

Dieter, U., and Ahrens, J. (1971). An exact determination of serial correlations of pseudo-random numbers. *Numer. Math.*, **17**, 101–123.

Disney, R. L. (1980). A tutorial on Markov renewal theory, semi-regenerative processes, and their applications. Technical Report, Virginia Polytechnic Institute and State University, Blacksburg, Va.

Downham, D. Y., and Roberts, F. D. K. (1967). Multiplicative congruential pseudo-random number generators. *Comput. J.*, **10**, 74–77.

Draper, N. R., and Smith, H. (1981). *Applied Regression Analysis*. Wiley: New York.

Duket, S. D., and Pritsker, A. A. B. (1978). Examination of simulation output using spectral methods. *Math. Comput. Simul.*, **20**, 53–60.

Dussault, J.-P. (1980). Réduction du biais des estimateurs de la variance dans une expérience avec variables de contrôle. Publication No. 389, Département d'informatique et de recherche opérationnelle, Université de Montréal.

Edwards, P. (1967). *The Encyclopedia of Philosophy*. Macmillan: New York.

Efron, B. (1982). *The Jackknife, the Bootstrap and other Resampling Plans*. (CBMS-NSF Series.) SIAM: Philadelphia.

Efron, B., and Stein, C. (1981). The jackknife estimate of variance. *Ann. Stat.*, **9**, 586–596.

Epstein, B. (1960). Testing for the validity of the assumption that the underlying distribution of life is exponential. *Technometrics*, **2**, 83–101.

Fama, E., and Roll, R. (1968). Some properties of symmetric stable distributions. *J. Amer. Stat. Assoc.*, **63**, 817–836.

Fama, E., and Roll, R. (1971). Parameter estimates for symmetric stable distributions. *J. Amer. Stat. Assoc.*, **66**, 331–338.

Fan, C. T., Muller, M. E., and Rezucha, I. (1962). Development of sampling plans by using sequential (item by item) selection techniques and digital computers. *JASA*, **57**, 387–402.

Federgruen, A., and Schweitzer, P. J. (1978). A survey of asymptotic value-iteration for undiscounted Markovian decision problems. Working Paper No. 7833, Graduate School of Management, University of Rochester.

Feller, W. (1971). *An Introduction to Probability Theory and Its Applications*, Vol. 2, 2nd edn. Wiley: New York.

Fishman, G. S. (1977). Achieving specific accuracy in simulation output analysis. *Commun. ACM*, **20**, 310–315.

Fishman, G. S. (1978). *Principles of Discrete Event Simulation*. Wiley: New York.

Fishman, G. S. (1981a). Accelerated accuracy in the simulation of Markov chains. Technical Report 81-1, Operations Research and Systems Analysis, University of North Carolina.

Fishman, G. S. (1981b). Accelerated convergence in the simulation of countably infinite state Markov chains. Technical Report 81-4, Operations Research and Systems Analysis, University of North Carolina.

Fishman, G. S., and Huang, B. D. (1980). Antithetic variates revisited. Technical Report 80-4, Curriculum in Operations Research and Systems Analysis, University of North Carolina, Chapel Hill.

Fishman, G. S., and Moore, L. R. (1978). A statistical analysis of multiplicative congruential generators with modulus $2^{31}-1$. Technical Report 78-11, Curriculum in Operations Research and Systems Analysis, University of North Carolina, Chapel Hill.

Ford, L. R., Jr., and Fulkerson, D. R. (1962). *Flows in Networks*. Princeton University Press: Princeton, N.J.

Forsythe, G. E. (1972). Von Neumann's comparison method for random sampling from the normal and other distributions. *Math. Comput.*, **26**, 817–826.

Fox, B. L. (1963). Generation of random samples from the beta and F distributions. *Technometrics*, **5**, 269–270.

Fox, B. L. (1968). Letter to the editor. *Technometrics*, **10**, 883–884.

Fox, B. L. (1978a). Estimation and simulation (communication to the editor). *Manage. Sci.* **24**, 860–861.

Fox, B. L. (1978b). Data structures and computer science techniques in operations research. *Oper. Res.*, **26**, 686–717.

Fox, B. L. (1981). Fitting 'standard' distributions to data is necessarily good: dogma or myth? *Proceedings 1981 Winter Simulation Conference*, pp. 305–307.

Fox, B. L. (1983a). Counterparts of variance reduction techniques for quasi-Monte Carlo integration. Manuscript, Université de Montréal.

Fox, B. L. (1983b). When conditional Monte Carlo works. Manuscript, Université de Montréal.

Franta, W. R. (1975). A note on random variate generators and antithetic sampling. *INFOR*, **13**, 112–117.

Franta, W. R. (1977). *The Process View of Simulation*. North Holland: New York.

Fulkerson, D. R. (1962). Expected critical path lengths in PERT networks. *Oper. Res.*, **10**, 808–817.

Gafarian, A. V., Ancker, C. J., Jr., and Morisaku, T. (1978). Evaluation of commonly used rules for detecting 'steady state' in computer simulation. *Nav. Res. Logistics Quart.*, **25**, 511–529.

Galambos, J. (1978). *The Asymptotic Theory of Extreme Order Statistics*. Wiley: New York.

Garey, M. R., and Johnson, D. S. (1979). *Computers and Intractability*. Freeman: San Francisco.

Gavish, B., and Merchant, D. K. (1978). Binary level testing of pseudo-random number generators. Technical Report, University of Rochester.

Gerontidis, I., and Smith, R. L. (1982). Monte Carlo generation of order statistics from general distributions. *Appl. Stat.* **31**, 238–243.

Glynn, P. W. (1982a). Some new results in regenerative process theory. Technical Report 16, Dept. of Operations Research, Stanford University, Stanford, Calif.

Glynn, P. W. (1982b). Regenerative aspects of the steady-state simulation problem for Markov chains. Technical Report 17, Dept. of Operations Research, Stanford University, Stanford, Calif.

Glynn, P. W. (1982c). Regenerative simulation of Harris recurrent Markov chains. Technical Report 18, Dept. of Operations Research, Stanford University, Stanford, Calif.

Glynn, P. W. (1982d). Coverage error for confidence intervals in simulation output analysis. Technical Report 15, Dept. of Operations Research, Stanford University, Stanford, Calif.

Glynn, P. W. (1982e). Asymptotic theory for nonparametric confidence intervals. Technical Report 19, Dept. of Operations Research, Stanford University, Stanford, Calif.

Gorenstein, S. (1967). Testing a random number generator. *Commun. ACM*, **10**, 111–118.

Granovsky, B. L. (1981). Optimal formulae of the conditional Monte Carlo. *SIAM J. Alg. Discrete Math.*, **2**, 289–294.

Grassmann, W. K. (1981). The optimal estimation of the expected number in a $M/D/\infty$ queueing system. *Oper. Res.*, **29**, 1208–1211.

Gray, H. L., and Schucany, W. R. (1972). *The Generalized Jackknife Statistic*. Dekker: New York.

Greenberger, M. (1961). Notes on a new pseudo-random number generator. *J. ACM*, **8**, 163–167.

Guibas, L. J., and Odlyzko, A. M. (1980). Long repetitive patterns in random sequences. *Z. Wahrsch.*, **53**, 241–262.

Gumbel, E. J. (1958). *Statistics of Extremes*. Columbia University Press: New York.

Gunther, F. L., and Wolff, R. W. (1980). The almost regenerative method for stochastic system simulations. *Oper. Res.*, **28**, 375–386.

Halfin, S. (1982). Linear estimators for a class of stationary queueing processes. *Oper. Res.*, **30**, 515–529.

Halton, J. (1970). A retrospective and prospective survey of the Monte Carlo method. *SIAM Rev.*, **12**, 1–63.

Hammersley, J. M., and Handscomb, D. C. (1964). *Monte Carlo Methods*. Methuen: London.

Hannan, E. J. (1970). *Multiple Time Series*. Wiley: New York.

Harris, C. M. (1976). A note on testing for exponentiality. *Nav. Res. Logistics Quart.*, **23**, 169–175.

Harrison, J. M., and Lemoine, A. J. (1977). Limit theorems for periodic queues. *J. Appl. Probab.*, **14**, 566–576.

Hart, J. F. *et al.* (1968). *Computer Approximations*. SIAM Series in Applied Mathematics, Wiley: New York.

Heidelberger, P., and Iglehart, D. L. (1979). Comparing stochastic systems using regenerative simulations with common random numbers. *Adv. Appl. Probab.*, **11**, 804–819.

Heidelberger, P., and Lewis, P. A. W. (1981). Regression-adjusted estimates for regenerative simulations, with graphics. *Commun. ACM*, **24**, 260–273.

Heidelberger, P., and Welch, P. D. (1981a). A spectral method for simulation confidence interval generation and run length control. *Commun. ACM*, **24**, 233–245.

Heidelberger, P., and Welch, P. D. (1981b). Adaptive spectral methods for simulation output analysis. *IBM J. Res. Devel.*, **25**, 860–876.

Heidelberger, P., and Welch, P. D. (1982). Simulation run length control in the presence of an initial transient. IBM Technical Report.

Henriksen, J. O. (1979). An interactive debugging facility for GPSS. *ACM SIGSIM Simuletter*, **10**, 60–67.

Heyman, D. P., and Stidham, Jr., S., (1980). The relation between customer and time averages in queues. *Oper. Res.*, **28**, 983–994.

Hill, G. W. (1970). Algorithm 395: Student's *t*-distribution. *Commun. ACM*, **13**, 617–619.

Hinderer, K. (1978). Approximate solutions of finite-state dynamic programs. In *Dynamic Programming and Its Applications* (M. L. Puterman, editor), pp. 289–317, Academic Press: New York.

Hoare, C. A. R. (1981). The emperor's old clothes. *Commun. ACM*, **24**, 75–83.

Hobson, E. W. (1957). *The Theory of Functions of a Real Variable*, Vol. 1. Dover: New York (reprint of the 1927 edition).

Hordijk, A., Iglehart, D. L., and Schassberger, R. (1976). Discrete-time methods of simulating continuous-time Markov chains. *Adv. Appl. Probab.*, **8**, 772–788.

IBM (1971). *General Purpose Simulation System V User's Manual*, 2nd edn. Publication SH20-0851-1. IBM: New York.

IBM (1972a). *SIMPL/1 Program Reference Manual.* Publication SH19-5060-0. IBM: New York.

IBM (1972b). *SIMPL/1 Operation Guide.* Publication SH19-5038-0. IBM: New York.

Iglehart, D. L. (1978). The regenerative method for simulation analysis. In *Current Trends in Programming Methodology—Software Modeling* (K. M. Chandy and R. T. Yeh, editors). Prentice-Hall: Englewood Cliffs, N.J.

Iglehart, D. L., and Lewis, P. A. W. (1979). Regenerative simulation with internal controls. *J. ACM*, **26**, 271–282.

Iglehart, D. L., and Shedler, G. S. (1983). Statistical efficiency of regenerative simulation methods for networks of queues. *Adv. Appl. Prob.* **15**, 183–197.

Iglehart, D. L., and Stone, M. L. (1982). Regenerative simulation for estimating extreme values. Technical Report 20, Dept. of Operations Research, Stanford University, Stanford, Calif.

Ignall, E. J., Kolesar, P., and Walker, W. E. (1978). Using simulation to develop and validate analytic models: some case studies. *Oper. Res.*, **26**, 237–253.

IMSL (1980). *IMSL Library*, Vol. I, 8th ed. Distributed by International Mathematical and Statistical Libraries, Inc., Houston, Tex.

Jensen, K., and Wirth, N. (1976). *PASCAL User Manual and Report*, 2nd ed., 2nd printing. Springer-Verlag: New York, Heidelberg, Berlin.

Jöhnk, M. D. (1964). Erzeugung von Betaverteilten und Gammaverteilten Zufallszahlen. *Metrika*, **8**, 5–15.

Karlin, S., and Taylor, H. M. (1975). *A First Course in Stochastic Processes*, 2nd ed. Academic Press: New York.

Katzan, H., Jr., (1971). *APL User's Guide.* Van Nostrand Reinhold: New York.

Kelton, W. D. (1980). The startup problem in discrete-event simulation. Report 80–1, Dept. of Industrial Engineering, University of Wisconsin, Madison.

Kennedy, Jr., W. J., and Gentle, J. E. (1980). *Statistical Computing.* Dekker: New York.

Kershenbaum, A., and Van Slyke, R. (1972). Computing minimum spanning trees efficiently. *Proceedings of the 25th ACM National Conference*, pp. 518–527.

Khintchine, A. Y. (1969). *Mathematical Methods in the Theory of Queueing.* Hafner: New York.

Kinderman, A. J., and Ramage, J. G. (1976). Computer generation of normal random variables. *JASA*, **71**, 893–896.

Kiviat, P. J., Villanueva, R., and Markowitz, H. M. (1975). *Simscript II.5 Programming Language*. CACI Inc.: Los Angeles.

Kleijnen, J. P. C. (1974). *Statistical Techniques in Simulation*, Part 1. Dekker: New York.

Kleijnen, J. P. C. (1975). *Statistical Techniques in Simulation*, Part 2. Dekker: New York.

Kleijnen, J. P. C. (1978). Communication to the editor (reply to Fox (1978a) and Schruben (1978)). *Manage. Sci.*, **24**, 1772–1774.

Kleinrock, L. (1975). *Queueing Systems*, Vol. 1. Wiley: New York.

Kleinrock, L. (1976). *Queueing Systems*, Vol. 2. Wiley: New York.

Knuth, D. E. (1973). *The Art of Computer Programming*, Vol. 1, 2nd ed. Addison-Wesley: Reading, Mass.

Knuth, D. E. (1975). Estimating the efficiency of backtrack programs. *Math. Comp.*, **29**, 121–136.

Knuth, D. E. (1981). *The Art of Computer Programming*, Vol. 2, 2nd ed. Addison-Wesley: Reading, Mass.

Koopman, B. O. (1977). Intuition in mathematical operations research. *Oper. Res.*, **25**, 189–206.

Koopman, B. O. (1979). An operational critique of detection. *Oper. Res.*, **27**, 115–133.

Kronmal, R. A., and Peterson, A. V. (1979). On the alias method for generating random variables from a discrete distribution. *Amer. Stat.*, **33**, 214–218.

Kronmal, R. A., and Peterson, A. V. (1981). A variant of the acceptance/rejection method for computer generation of random variables. *JASA*, **76**, 446–451.

Landauer, J. P. (1976). Hybrid digital/analog computer systems. *Computer*, **9**, 15–24.

Landwehr, C. E. (1980). An abstract type for statistics collection in SIMULA. *ACM Trans. Prog. Lang. Syst.*, **2**, 544–563.

Larmouth, J. (1981). Fortran 77 portability. *Software-Pract. Exper.*, **11**, 1071–1117.

Lavenberg, S. S., Moeller, T. L., and Welch, P. D. (1982). Statistical results on multiple control variables with application to queueing network simulation. *Oper. Res.*, **30**, 182–202.

Lavenberg, S. S., and Sauer, C. H. (1977). Sequential stopping rules for the regenerative method of simulation. *IBM J. Res. Develop.*, **21**, 545–558.

Lavenberg, S. S., and Welch, P. D. (1981). A perspective on the use of control variables to increase the efficiency of Monte Carlo simulations. *Manage. Sci.*, **27**, 322–335.

Law, A. M. (1975). Efficient estimators for simulated queueing systems. *Manage. Sci.*, **22**, 30–41.

Law, A. M. (1977). Confidence intervals in discrete-event simulation: a comparison of replication and batch means. *Nav. Res. Logistics Quart.*, **24**, 667–678.

Law, A. M. (1979). *Statistical Analysis of Simulation Output Data With Simscript II.5*. CACI Inc.: Los Angeles.

Law, A. M., and Carson, J. S. (1979). A sequential procedure for determining the length of a steady-state simulation. *Oper. Res.*, **27**, 1011–1025.

Law, A. M., and Kelton, W. D. (1979). Confidence intervals for steady-state simulation, I: A survey of fixed sample size procedures. Report 78–5, Dept. of Industrial Engineering, University of Wisconsin, Madison.

Law, A. M., and Kelton, W. D. (1982). Confidence intervals for steady-state simulation, II: A survey of sequential procedures. *Manage. Sci.*, **28**, 550–562.

Lee, Y. T., and Requicha, A. A. G. (1982a). Algorithms for computing the volume and other integral properties of solids, I: Known methods and open issues. *Commun. ACM*, **25**, 635–641.

Lee, Y. T., and Requicha, A. A. G. (1982b). Algorithms for computing the volume and other integral properties of solids, II: A family of algorithms based on representation conversion and cellular approximation. *Commun. ACM*, **25**, 642–650.

Lehmann, E. L. (1966). Some concepts of dependence. *Ann. Math. Stat.*, **37**, 1137–1153.

Leringe, Ö., and Sundblad, Y. (1975). *Simula för den som kan Algol* (English translation: "Simula for those who know Algol"). Tekniska Högskolans Studentkår Kompendieförmedlingen: Stockholm.

Lewis, P. A. W., and Shedler, G. S. (1979a). Simulation of nonhomogeneous Poisson processes by thinning. *Nav. Res. Logistics Quart.*, **26**, 403–414.

Lewis, P. A. W., and Shedler, G. S. (1979b). Simulation of nonhomogeneous Poisson processes with degree-two exponential polynomial rate function. *Oper. Res.*, **27**, 1026–1040.

Lewis, T. G., and Payne, W. H. (1973). Generalized feedback shift register pseudo-random number algorithm. *J. ACM*, **20**, 456–468.

Lilliefors, H. W. (1967). On the Kolmogorov–Smirnov test for normality with mean and variance unknown. *JASA*, **62**, 399–402.

Lilliefors, H. W. (1969). On the Kolmogorov–Smirnov test for the exponential distribution with mean unknown. *JASA*, **64**, 387–389.

Ling, R. E., and Roberts, H. V. (1980). *User's Manual for IDA*. Scientific Press: Palo Alto, Calif.

Lurie, D., and Hartley, H. O. (1972). Machine-generation of order statistics for Monte Carlo computations. *Amer. Stat.*, **26**, 26–27.

McCormack, W. M., and Sargent, R. G. (1981). Analysis of future event set algorithms for discrete-event simulation. *Commun. ACM*, **24**, 801–812.

McKeon, R. (1941). *The Basic Works of Aristotle*. Random House: New York.

MacLaren, M. D., Marsaglia, G., and Bray, T. A. (1964). A fast procedure for generating exponential random variables. *Commun. ACM*, **7**, 298–300.

MacWilliams, F. J., and Sloane, N. J. A. (1976). Pseudo-random sequences and arrays. *Proc. IEEE*, **64**, 1715–1729.

Mahl, R., and Boussard, J. C. (1977). *Algorithmique et structures de données*. Laboratoire d'informatique, Université de Nice: Nice.

Mann, H. B., and Wald, A. (1942). On the choice of the number of class intervals in the application of the chi-square test. *Ann. Math. Stat.*, **13**, 306–317.

Marsaglia, G. (1961). Expressing a random variable in terms of uniform random variables. *Ann. Math. Stat.*, **32**, 894–899.

Marsaglia, G. (1964). Generating a variable from the tail of the normal distribution. *Technometrics*, **6**, 101–102.

Marsaglia, G. (1968). Random numbers fall mainly in the planes. *Proc. Nat. Acad. Sci.*, **60**, 25–28.

Marsaglia, G. (1977). The squeeze method for generating gamma variates. *Comput. Math. Appl.*, **3**, 321–325.

Marsaglia, G., and Bray, T. A. (1968). On-line random number generators and their use in combinations. *Commun. ACM*, **11**, 757–759.

Marshall, A. W. (1956). In *Symposium on Monte Carlo Methods* (H. A. Meyer, editor), p. 14. Wiley: New York.

Meilijson, I., and Nádas, A. (1979). Convex majorization with an application to the length of critical paths. *J. Appl. Probab.*, 671–677.

Meketon, M. S., and Heidelberger, P. (1982). A renewal theoretic approach to bias reduction in regenerative simulations. *Manage. Sci.*, **26**, 173–181.

Melamed, B. (1979). Characterizations of Poisson traffic streams in Jackson queueing networks. *Adv. Appl. Probab.*, **11**, 422–438.

Melamed, B. (1982). On Markov jump processes imbedded at jump epochs and their queueing-theoretic applications. *Math. Oper. Res.*, **7**, 111–128.

Miller, R. G. (1974). The jackknife—a review. *Biometrika*, **61**, 1–5.

Miller, R. G. (1981). *Simultaneous Statistical Inference*. Springer-Verlag: New York, Heidelberg, Berlin.

Mitchell, B. (1973). Variance reduction by antithetic variates in $G/G/1$ queueing simulations. *Oper. Res.*, **21**, 988–997.

Morse, P. M. (1977). ORSA twenty-five years later. *Oper. Res.*, **25**, 186–188.

Mulvey, J. M. (1980). Reducing the U.S. Treasury's taxpayer data base by optimization. *Interfaces*, **10**, 101–112.

Nance, R. E., and Overstreet, Jr., C., (1978). Some observations on the behavior of composite random number generators. *Oper. Res.*, **26**, 915–935.

Neuts, M. F. (editor) (1977a). *Algorithmic Methods in Probability* (TIMS Studies in the Management Sciences), Vol. 7. North Holland: Amsterdam.

Neuts, M. F. (1977b). The mythology of steady state. Working Paper, Dept. of Statistics and Computer Science, University of Delaware, Newark. Presented at the ORSA-TIMS Meeting, Atlanta, Ga., November 7–9, 1977.

Niederreiter, H. (1978). Quasi-Monte Carlo methods and pseudo-random numbers. *Bull. Amer. Math. Soc.*, **84**, 957–1042.

Odeh, R. E., and Evans, J. O. (1974). Algorithm AS 70: percentage points of the normal distribution, *Appl. Stat.*, **23**, 96–97.

Ohlin, M. (1977). Next random—a method of fast access to any number in the random generator cycle. *Simula Newslett.*, **6**, 18–20.

Parzen, E. (1962). *Stochastic Processes.* Holden-Day: San Francisco.

Paulson, A. S., Holcomb, E. W., and Leitch, R. A. (1975). The estimation of parameters of the stable laws. *Biometrika*, **62**, 163–170.

Payne, W. H. (1970). FORTRAN Tausworthe pseudorandom number generator. *Commun. ACM*, **13**, 57.

Payne, W. H., Rabung, J. R., and Bogyo, T. P. (1969). Coding the Lehmer pseudo-random number generator. *Commun. ACM*, **12**, 85–86.

Pike, M. C., and Hill, I. D. (1966). Algorithm 291—logarithm of gamma function. *Commun. ACM*, **9**, 684–685.

Pritsker, A. A. B. (1974). *The GASP IV Simulation Language.* Wiley: New York.

Pritsker, A. A. B. (1977). *Modeling and Analysis Using Q-GERT Networks.* Halsted Press (a division of Wiley): New York.

Pritsker, A. A. B., and Kiviat, P. J. (1969). *Simulation with Gasp II: A Fortran Based Simulation Language.* Prentice-Hall: Englewood Cliffs, N.J.

Pritsker, A. A. B., and Pegden, C. D. (1979). *Introduction to Simulation and SLAM.* Systems Publishing Corp.: W. Lafayette, Ind.

Proceedings of the Winter Simulation Conference (1981) (available from: Association for Computing Machinery, 1133 Avenue of the Americas, New York, N.Y. 10036).

Purdom, P. W. (1978). Tree size by partial backtracking. *SIAM J. Comput.*, **7**, 481–491.

Ramberg, J., and Schmeiser, B. (1974). An approximate method for generating asymmetric random variables. *Commun. ACM*, **17**, 78–82.

Rand Corporation (1955). *A Million Random Digits with 100,000 Normal Deviates.* Free Press: Glencoe, Ill.

Renyi, A. (1953). On the theory of order statistics. *Acta Math. Acad. Sci. Hung.*, **4**, 191–231.

Rosenblatt, M. (1975). Multiply schemes and shuffling. *Math. Comp.*, **29**, 929–934.

Ross, S. M. (1970). *Applied Probability Models with Optimization Applications.* Holden-Day: San Francisco.

Rothery, P. (1982). The use of control variates in Monte Carlo estimation of power. *Appl. Stat.*, **31**, 125–129.

Rothkopf, M. (1969). A model of rational competitive bidding. *Manage. Sci.*, **15**, 362–373.

Rothkopf, M., and Oren, S. S. (1979). A closure approximation for the nonstationary $M/M/s$ queue. *Manage. Sci.*, **25**, 522–534.

Russell, E. C. (1982). *Building Simulation Models with Simscript II.5*, preliminary printing. CACI Inc.: Los Angeles.

Sargent, R. G. (1980). Verification and validation of simulation models. Working Paper No. 80-013, College of Engineering, Syracuse University, Syracuse, New York.

Scheffé, H. (1959). *The Analysis of Variance.* Wiley: New York.

Schmeiser, B. (1980). Generation of variates from distribution tails. *Oper. Res.*, **28**, 1012–1017.

Schmeiser, B. (1982). Batch size effects in the analysis of simulation output. *Oper. Res.*, **30**, 556–568.

Schmeiser, B., and Babu, A. J. G. (1980). Beta variate generation via exponential majorizing functions. *Oper. Res.*, **28**, 917–926.

Schmeiser, B., and Kachitvichyanukul, V. (1981). Poisson random variate generation. Research Memorandum 81–4, School of Industrial Engineering, Purdue University, West Lafayette, Ind.

Schmeiser, B., and Lal, R. (1980). Squeeze methods for generating gamma variates. *JASA*, **75**, 679–682.

Schmeiser, B., and Lal, R. (1982). Bivariate gamma random vectors. *Oper. Res.*, **30**, 355–374.

Schrage, L. (1979). A more portable Fortran random number generator. *ACM Trans. Math. Software*, **5**, 132–138.

Schriber, T. J. (1974). *Simulation Using GPSS*. Wiley: New York.

Schruben, L. W. (1978). Communication to the editor (reply to Fox (1978a)). *Manage. Sci.*, **24**, 862.

Schruben, L. W. (1981). Analysis of simulation event incidence graphs. Technical Report 498, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, N.Y.

Schruben, L. W. (1982a). Detecting initialization bias in simulation output. *Oper. Res.*, **30**, 569–590.

Schruben, L. W. (1982b). Confidence interval estimation from standardized simulation output. Technical Report 518, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, N.Y.

Schruben, L. W., and Margolin, B. H. (1978). Pseudorandom number assignment in statistically designed simulation and distribution sampling experiments. *JASA*, **73**, 504–525.

Schucany, W. R., (1972). Order statistics in simulation. *J. Stat. Comp. Simul.*, **1**, 281–286.

Shapiro, S. S., and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, **52**, 591–611.

Shapiro, S. S., Wilk, M. B., and Chen, H. J. (1968). A comparative study of various tests of normality. *JASA*, **63**, 1343–1372.

Shearn, D. C. S. (1975). Discrete event simulation in Algol 68. *Software—Pract. Exper.*, **5**, 279–293.

Sheil, B. A. (1981). The psychological study of programming. *Comput. Surv.*, **13**, 101–120.

Siegmund, D. (1976). Importance sampling in the Monte Carlo study of sequential tests. *Ann. Stat.*, **4**, 673–684.

Siklósi, K. (1975). *Simula Simulation*. Tekniska Högskolans Studentkår Kompendieförmedlingen: Stockholm.

Solis, F. J., and Wets, R. J.-B. (1981). Minimization by random search techniques. *Math. Oper. Res.*, **6**, 19–30.

Stahnke, W. (1973). Primitive binary polynomials. *Math. Comp.*, **27**, 977–980.

Starr, N. (1966). The performance of a sequential procedure for the fixed-width interval estimation of the mean. *Ann. Math. Stat.*, **37**, 36–50.

Stidham, Jr., S., (1970). On the optimality of single-server queueing systems. *Oper. Res.*, **18**, 708–732.

Strauch, R. E. (1970). When a queue looks the same to an arriving customer as to an observer. *Manage. Sci.*, **17**, 140–141.

Sunder, S. (1978). Personal communication.

Tadikamalla, P. R. (1978). Computer generation of gamma random variables, II. *Commun. ACM*, **21**, 925–927.

Tadikamalla, P. R. (1980). Random sampling from the exponential power distribution. *JASA*, **75**, 683–686.

Tadikamalla, P. R., and Johnson, M. E. (1981). A complete guide to gamma variate generation. *Amer. J. Math. Manage. Sci.*, **1**, 213–236.

Tannenbaum, A. S. (1981). *Computer Networks.* Prentice-Hall: Englewood Cliffs, N.J.

Tausworthe, R. C. (1965). Random numbers generated by linear recurrence modulo two. *Math. Comput.*, **19**, 201–209.

Theil, H. (1961). *Economic Forecasts and Policy.* North-Holland: Amsterdam.

Thesen, A. (1977). The evolution of a new discrete event simulation language for inexperienced users (WIDES). *Software—Pract. Exper.*, **7**, 519–533.

Toothill, J. P. R., Robinson, W. D., and Adams, A. G. (1971). The runs up-and-down performance of Tausworthe pseudo-random number generators. *J. ACM*, **18**, 381–399.

Van Slyke, R. (1963). Monte Carlo methods and the PERT problem. *Oper. Res.*, **11**, 839–860.

Vaucher, J. G. (1973). La programmation avec Simula 67. Document de travail no. 36, Département d'informatique et de recherche opérationnelle, Université de Montréal, Montréal.

Vaucher, J. G. (1975). Prefixed procedures: a structuring concept for operations. *INFOR*, **13**, 287–295.

Vaucher, J. G. (1976). On the distribution of event times for the notices in a simulation event list. *INFOR*, **15**, 171–182.

Vaucher, J. G., and Duval, P. (1975). A comparison of simulation event list algorithms. *Commun. ACM*, **18**, 223–230.

Von Neumann, J. (1951). Various techniques used in connection with random digits. *National Bureau of Standards Applied Mathematics*, Series 12, pp. 36–38.

Wagner, H. M. (1975). *Principles of Operations Research*, 2nd ed. Prentice-Hall: Englewood Cliffs, N.J.

Wahba, G. (1980). Automatic smoothing of the log periodogram. *JASA*, **75**, 122–132.

Walker, A. J. (1977). An efficient method for generating discrete random variables with general distributions. *ACM Trans. Math. Software*, **3**, 253–256.

Weissman, I. (1978a). Estimation of parameters and large quantiles based on the k largest observations. *JASA*, **73**, 812–815.

Weissman, I. (1978b). Private communication.

Welsh, J., and McKeag, M. (1980). *Structured System Programming.* Prentice-Hall: Englewood Cliffs, N.J.

West, J. (1979). *SIMGRAPH—A Graphics Package for Simscript II.5.* CACI Inc.: Los Angeles.

Whitt, W. (1976). Bivariate distributions with given marginals. *Ann. Stat.*, **4**, 1280–1289.

Whitt, W. (1983). Untold horrors of the waiting room: what the steady-state distribution will never tell about the queue-length process. *Manage. Sci.* (to appear).

Wichman, B. A., and Hill, I. D. (1982). An efficient and portable pseudo-random number generator. *Appl. Stat.*, **31**, 188–190.

Wolff, R. W. (1982). Poisson arrivals see time averages. *Oper. Res.*, **30**, 223–231.

Woolsey, R. E. D., and Swanson, H. S. (1975). *Operations Research for Immediate Applications.* Harper & Row: New York.

Wright, R. D., and Ramsay, Jr., T. E. (1979). On the effectiveness of common random numbers. *Manage. Sci.*, **25**, 649–656.

Xerox (1972). *SL-1 Reference Manual.* Publication 90 16 76B, Xerox Corporation, El Segundo, Calif.

Yakowitz, S., Krimmel, J. E., and Szidarovszky, F. (1978). Weighted Monte Carlo integration. *SIAM J. Numer. Anal.*, **15**, 1289-1300.

Yuen, C. (1982). The inversion method in random variate generation. M.Sc. Thesis, School of Computer Science, McGill University, Montreal.

Zierler, N. (1969). Primitive trinomials whose degree is a Mersenne exponent. *Inf. Cont.*, **15**, 67-69.

Author Index

Subject Index