Pseudorandom number generators revisited

Kleijnen, J.P.C.; Annink, B.

Publication date:
1989

TILBURG UNIVERSITY
KATHOLIEKE UNIVERSITEIT BRABANT
POSTBOX 90153
5000 LE TILBURG
THE NETHERLANDS

[redacted]

DEPARTMENT OF ECONOMICS

RESEARCH MEMORANDUM

PSEUDORANDOM NUMBER GENERATORS REVISITED

Jack P.C. Kleijnen
Ben Annink

FEW 388

PSEUDORANDOM NUMBER GENERATORS REVISITED

Jack P.C. Kleijnen
and
Ben Annink

May 1989

Correspondence should be directed to: Prof. J.P.C. Kleijnen, Department of Information Systems and Auditing, School of Business and Economics, Catholic University Brabant (Katholieke Universiteit Brabant), 5000 LE Tilburg, Netherlands. FAX: 013-663019. E-mail: t435klei@htikub5.

PSEUDORANDOM NUMBER GENERATORS REVISITED

Jack P.C. KLEIJNEN and Ben ANNINK
Department of Information Systems and Auditing
School of Business and Economics
Catholic University Brabant (Katholieke Universiteit Brabant)
P.O. Box 90153
5000 LE Tilburg, Netherlands

When splitting the cycle (of length h) of a multiplicative generator into two parts, the pseudorandom numbers across parts (x_i and $x_{i+h/2}$ with $i = 1,\ldots,h/2-1$) turn out to lie on only two parallel lines. These "long range" correlations have consequences for classic and for parallel computers. For supercomputers simple alternative generators are presented. These generators are more efficient than the standard subroutines (RANF and VRANF) available on the CYBER 205.

Simulation, Monte Carlo, parallel algorithms, random number generation, software

1. Introduction

Is there really a need for any more research on pseudorandom number generators? Isn't it like beating a dead horse? But no: new properties of old generators are still discovered, and new generators must be developed to accommodate new architectures of computers, especially pipelined supercomputers and parallel processors, as we shall see. Recently some interesting results on pseudorandom number generators have appeared; unfortunately these results have been published scattered over various journals that are not easily accessible to readers of this journal, we think.

We concentrate on one class of generators, namely linear congruential generators. These generators are most popular; a recent critical survey is Park and Miller (1988).

This paper is organized as follows. In § 2 we summarize basic results for linear congruential generators that we need in the sequel. In § 3 we split up the full cycle of the multiplicative generator into equal parts, first into two parts (§ 3.1), then into 2^k parts (§ 3.2), showing that the pseudorandom numbers lie on two and on no more than 2^{k-1} parallel lines if $k \le 2$ and $k \ge 3$ respectively. We briefly consider antithetic pseudorandom numbers in § 3.3, and in § 3.4 we study the conditional variances and the correlation coefficient of the pseudorandom numbers paired across two parts (of the 2^k parts). Finally § 3.5 summarizes the disadvantages of splitting a pseudorandom number stream into parts. Therefore § 4 gives alternative generators for supercomputers. First we briefly explain the 'assembly line' architecture of supercomputers such as the CYBER 205. Next § 4.1 gives one situation requiring computation of J multipliers ($a_j = a^j \bmod m$), and § 4.2 gives a related parallel algorithm requiring computation of a single multiplier ($a^J \bmod m$) and initializing a vector with J successive numbers. Finally § 4.3 compares these two generators to the standard scalar routine RANF and the standard vector routine VRANF on the CYBER 205. At the end § 5 summarizes conclusions.

2. Linear Congruential Generators

Linear congruential generators have the form

$$x_{j+1} = (a\, x_j + c) \bmod m \qquad (j = 0,1,2,\ldots), \tag{2.1}$$

where the multiplier a, the constant c, the modulus m, and the seed x_0 are integers. When c is zero, the generator is called multiplicative congruential. The generator has a specific cycle length or period h; which means that if the generator starts with seed x_0 then $x_h = x_0$ so that $x_{h+1} = x_1$ and so on. Obviously for the pseudorandom numbers $r_j = x_j/m$ we have $0 \le r_j < 1$. An efficient algorithm results if $m = 2^w$ where w depends on the computer's word size; for example, CDC's supercomputer CYBER 205 uses $m = 2^{47}$ (see CDC, 1986), but IMSL uses $m = 2^{31}-1$ for 'classic' computers; NAG uses $m = 2^{59}$ (double word on classic computers). However, there are other considerations than efficiency.

Pseudorandom number generators should yield results r_j that can be considered as statistically independent; that is, the observed sequence $r_0, r_1, \ldots r_n$ should not provide any information about the next sequence $r_{n+1}, r_{n+2}, \ldots$ It is extremely difficult to meet this requirement (see

Bratley et al., 1983; Fishman, 1978; Park and Miller, 1988; Ripley, 1987). It is possible to derive conditions that are necessary but not sufficient. For example, the following lemma is well known.

Lemma 1: If in equation (2.1) $m = 2^w$ and $c = 0$, then the maximum cycle length is $h = 2^{w-2}$; this maximum is reached if $a = 4g \pm 1$ with odd integer g.

Because these conditions are not sufficient, we should apply statistical tests to the generator's output (r_0, r_1, ...) to see if several types of statistical dependence are absent indeed. For example, two-tuples (r_0, r_1), (r_2, r_3), (r_4, r_5) ... should be uniformly distributed over the unit square; Figure 1 shows results for a pedagogical example that can be easily checked by the reader. We shall return to this figure.

3. Partitioning the Cycle

Kleijnen (1989) surveys several types of linear consequential generators for supercomputers. In § 4 we shall discuss supercomputers; now we mention only that Kleijnen (1989) discusses splitting up the cycle of pseudorandom numbers into 65, 535 (= 2^{16}-1) non-overlapping parts. We now prove that this approach is wrong! This prove reveals properties of generators also of interest for classic computers. In this section we restrict ourselves to multiplicative generators with $m = 2^w$, a multiplier a selected such that $h = 2^{w-2}$, and a seed $x_0 = 1$; see lemma 1. First we consider partitioning into only two parts, next into more parts.

3.1. Partitioning into two parts

Suppose we split the cycle of length $h = 2^{w-2}$ into two equal parts of length $h/2$. Kleijnen (1989) assumes that the generator is tested on a classic computer; more specifically, he assumes that the pseudorandom numbers r_j ($j = 0,\ldots,h$) are statistically independent. Unfortunately, the numbers r_j, or equivalently x_j, are statistically dependent. More specifically, De Matteis and Pagnutti (1988) give number theoretic results which guide our present research.

Let us return to the pedagogical example of Figure 1(b) with $m = 2^6$ and $h = 2^{6-2} = 16$. Splitting into two parts yields a first part consisting of $x_0, x_1, \ldots, x_7$; the second part comprises $x_8, x_9, \ldots, x_{15}$. Now we plot the pairs corresponding across the two parts: (x_0, x_8), (x_1, x_9), $\ldots$, (x_7, x_{15}). So we are interested, not in first-order autocorrelation (Figure 1), but in long range correlation. This yields Figure 2.

A more realistic generator has a higher modulus m and hence a longer cycle h. We present plots only for $m = 2^{12}$ and $a = 5$, which are easily obtained on a Personal Computer. In Lemma 2, however, we shall see that the pattern shown by these plots holds for all generators considered in this section. Figure 3 shows the plot for partitioning into two parts: $(x_0, x_{h/2})$, $(x_1, x_{h/2+1})$, $\ldots$, $(x_{h/2-1}, x_h)$. Again all these h/2 pairs lie on only two parallel lines with slope one; these lines have no overlapping domains; a small number in the first part $(0 < r_j < 0.5)$ goes together with a high number in the second part $(0.5 < r_{h/2+j} < 1)$. (So the pseudo-random numbers are negatively correlated; see Table 2 later on.) In figure 3 we display r, not x, in order to make the plots independent of m.

3.2. Partitioning into 2^k parts

What happens if we double the number of parts? First, we should notice the relationship between partitioning into two and four equal parts respectively. Let us return to the didactic example with $m = 2^6$. When we splitted the cycle into two parts, we plotted (x_0, x_8), (x_1, x_9), $\ldots$, (x_7, x_{15}). Now we have four parts, each of length $h = 2^{w-2}/4 = 2^{6-2}/2^2 = 4$, namely part #1 is (x_0, x_1, x_2, x_3), part #2 is (x_4, x_5, x_6, x_7), part #3 is $(x_8, x_9, x_{10}, x_{11})$, and part #4 is $(x_{12}, x_{13}, x_{14}, x_{15})$. So the pairs across parts #1 and #3 are: (x_0, x_8), (x_1, x_9), (x_2, x_{10}), (x_3, x_{11}). But these four pairs also occured in the plot for two parts! So if if splitting up into two parts gives unacceptable results, then splitting up into four parts and using all parts does not help! We must split up the cycle into more parts and <u>use the first two parts only</u>. Figure 4 displays the plot for parts #1 and #2: $(x_0, x_{h/4})$, $(x_1, x_{h/4+1})$, $\ldots$, $(x_{h/4-1}, x_{h/2-1})$. Again all h/4 pairs lie on only two parallel lines with slope one; these lines still have no overlapping domains; compared to splitting up into only two parts (Figure 3) these lines shifted to the left (the correlation is still negative but smaller in absolute magnitude; see Table 2).

The pattern of the plots changes as we go on doubling the number of equal parts! Figure 5 gives the plot for the first two parts when splitting up into 2^3 parts: $(x_0, x_{h/8})$, $(x_1, x_{h/8+1})$, ..., $(x_{h/8-1}, x_{h/4-1})$. Again all h/8 pairs lie on parallel lines with slope one, but there are now four lines and some of these lines have partially overlapping domains; a small number in the first part 'goes together' with two different values in the second part (strictly speaking, one particular value of x_j corresponds to a unique value for $x_{h/8+j}$ since all numbers x are different in a multiplicative generator; we shall return to this issue).

Figure 6 plots the pairs when splitting up into 16 parts. Again all h/16 pairs lie on parallel lines with slope one, but there are now eight such lines with more overlap of domains. Figure 7 gives results for 32 parts. All h/32 pairs still lie on parallel lines with slope one, but there are now more such lines, even though these lines are now hard to distinguish because there are few points per line. And so we could continue. Actually De Matteis and Pagnutti (1988, p. 604) prove the following.

Lemma 2: Suppose the modulus of the multiplicative genetor is $m = 2^w$ with $w > 3$, the multiplier a is chosen such that the cycle length is $h = 2^{w-2}$, and the seed is $x_0 = 1$. Divide the resulting sequence into 2^k parts with $k < w-2$. If $k \leq 2$ then x_j and $x_{j+h/(2^k)}$ lie on two parallel lines with slope one. If $k > 2$ then there are no more than 2^{k-1} parallel lines.

3.3. Antithetic pseudorandom numbers

Kleijnen (1974, p. 254) proves that the antithetic pseudorandom numbers $1-r_j$ can be generated by starting with the seed $m-x_0$. Hence the antithetic numbers (say) y_j satisfy $y_j = m-x_j$ for j = 1, 2, ..., h-1, h. Combined with Lemma 2, this means that the cycle of the antithetic numbers y_j *has no element in common* with the cycle of the 'original' numbers x_j.

Note that a multiplicative generator with $m = 2^w$ has a maximum cycle of length $h = 2^{w-2}$; see Lemma 1. This can be explained as follows.

The modulus $m = 2^w$ results in odd values only: half the cycle running from 0 through m-1 is lost that way. Another half lies in the antithetic cycle.

3.4. Statistical analysis

The preceding plots illustrate number theoretic results. What are the statistical consequences? First we see that, within the cycle, no number x_j occurs more than once, whereas the statistical analysis of simulation output assumes that random numbers are sampled independently and hence specific values can occur more than once. In the statistical analysis this phenomenon is always ignored. Analogously in our analysis of the preceding figures we assume continuous parallel equidistant lines in the unity quadrant. We assume that the generator does yield a uniform marginal distribution; hence var(r) = 1/12. It is easy to derive the variance of $r_{j+h/(2^k)}$ given r_j and given a partitioning of the cycle into 2^k parts ($j = 0,\ldots,h/(2^k)-1$). For example, for k = 3 Figure 5 gives four lines such that with each r_j two values for $r_{h/8+j}$ correspond. We assume that these two values are equally probable. Obviously the distance between two neighboring lines is 1/2. Hence

$$\mathrm{var}(r_{h/8+j}|r_j) = \{(1/4)^2 + (1/4)^2\}1/2 = 1/16.$$

This yields Table 1. This Table shows that the conditional variance increases monotonically to 1/12, the variance if the second part would be independent of the first part.

Table 1: Conditional variance of $r_{h/(2^k)+j}$ given r_j for 2^k parts as a percentage of $\mathrm{var}(r_{h/(2^k)+j}) = 1/12$.

k	1	2	3	4	5	6	7
$\mathrm{var}(r_{h/(2^k)+j}\|r_j)$	0	0	75%	93.75%	98.44%	99.61%	99.90%

We also test the correlation coefficient between the pairs $(r_j, r_{h/(2^k)+j})$. If the r were normally distributed then zero correlation would imply independence. In case of nonnormality this is not true; for example, if

$$r_{h/(2^k)+j} = r_j \quad \text{for } 0 < r_j < 0.5$$
$$= 1-r_j \quad \text{for } 0.5 < r_j < 1, \tag{3.2}$$

then their correlation is zero. To test for zero correlation of the uniformly distributed r we use the "Spearman rank correlation test"; see Churchill (1983, pp. 596-598). So if the rank of r_j is v_j and that of $r_{h/(2^k)+j}$ is $w_{h/(2^k)+j}$, then we compute

$$R = 1 - \frac{\sum_{j=1}^{n} (v_j - w_{h/(2^k)+j})^2}{n(n^2-1)} \tag{3.3}$$

Obviously max(R) = 1. The following statistic has the t distribution with n-2 degrees of freedom:

$$T = \frac{R(n-2)^{1/2}}{1-R^2} \tag{3.4}$$

We compute T for n = 1000 and a popular generator, namely $m = 2^{32}$ and a = 69069. This yields Table 2.

Table 2: Spearman rank correlation test for $(r_{h/(2^k)+j}, r_j)$ when partitioning the cycle into 2^k parts; $m = 2^{32}$ and a = 69069; n = 1000.

k =	1	2	3	4	5
T =	-17.94	-4.56	-1.05	0.68	-0.19

This table gives a nonsignificant correlation for k = 3. Nevertheless Figure 5 and Table 1 suggest a strong dependence; also see the example in equation (3.2).

If pseudorandom numbers are dependent then the simulation fed by these numbers, gives dependent results. The statistical analysis of the simulation output assumes independence when estimating variances and confidence intervals!

3.5. Summary of splitting approach

Kleijnen (1989) assumes that the pseudorandom numbers r_j are truly independent. Then it makes sense to generate (say) J numbers in parallel by selecting J seeds such that the full cycle is split up into J equal parts. Number theoretical results derived by De Matteis and Pagnutti (1988), however, prove that these parts may be correlated, especially if J is small. Acceptable statistical behavior requires that the cycle be split into at least 2^5 parts and that only the first two parts be used. So of the full cycle of length $h = 2^{w-2}$ we use only $2 \times 2^{w-2-5}$ numbers. The useful part is split into J subparts for parallel generation of pseudorandom numbers; see Kleijnen (1989). We emphasize that the long range correlation also causes problems on classic computers if relatively many pseudorandom numbers are needed. In § 4 we shall present generators for vector computers that produce pseudorandom numbers <u>not</u> spread over the full cycle (with the concomittant problem of long range correlation). Moreover, these generators produce numbers in exactly the same order as generators for classic computers do; this facilitates debugging.

4. Pipeline computers and generators

First we consider the <u>pipeline</u> architecture of supercomputers such as the CYBER 205. We start with an example, namely the innerproduct of two vectors, $\underset{\sim}{v}_1\ \underset{\sim}{v}_2 = \sum_1^J v_{1j}\ v_{2j}$. This computation requires J scalar multiplications $v_{1j}\ v_{2j}$; these J operations can be done in parallel because the product $v_{1j}\ v_{2j}$ does not need the product $v_{1(j-1)}\ v_{2(j-1)}$. The pipeline architecture means that the computer works as an assembly line; that is, efficiency improves drastically if a large number of identical operations can be executed, independently of each other; see Levine (1982). Such supercomputers are efficient, only if these operations can be executed independently or in parallel, which means that <u>recursive</u> statements are not suited to pipelined computers. Unfortunately, the linear congruential generator is recursive: equation (2.1) shows that to compute x_{j+1} its predecessor x_j is needed. Moreover, because of fixed set-up costs, the 'assembly line' computer is efficient only if the number of basic operations is large; the literature suggests $J > 50$. There is also a technical upper limit on J, namely $J \le 2^{16}-1 = 65{,}535$ because the CYBER 205 uses 16 bits for addressing; see SARA (1984, p. 26). So the computer generates J pseudorandom numbers in parallel with $50 < J \le 65{,}535$. Hence a simulation experiment that requires N pseudorandom numbers, must call this parallel routine $[N/J]$ times where [ ] denotes rounding upwards to the next integer; for example, if $N = 1{,}000{,}000$ and $J = 65{,}535$ then 16 calls are necessary. So we may imagine an $I \times J$ matrix of pseudorandom numbers, where J numbers are generated in parallel and I calls are made to that vector routine. Kleijnen (1989) surveys different solutions to this problem (namely, J different multipliers m_j and J additive constants c_j; sampling J seeds; selecting J seeds I apart; also see § 3). He rejected the following idea because of overflow problems, but we shall show how to solve this problem.

4.1. Vector of multipliers

Fishman (1978) proves that, given a seed x_0 and J calls to the classic multiplicative generator (see equation 2.1 with $c = 0$), the resulting number x_J can be derived without knowing the intermediate numbers $(x_1, x_2, \ldots, x_{J-1})$:

$$x_J = (a^J x_0) \bmod m. \tag{4.1}$$

So we can generate J pseudorandom numbers in parallel if we first generate, once and for all, the vector of J multipliers: $\underset{\sim}{a} = (a_1, a_2, \ldots, a_{J-1}, a_J)'$ with

$$a_j = (a^j) \bmod m \qquad j = 1, \ldots, J. \tag{4.2}$$

This vector is multiplied by the scalar x_0 to give the vector $(x_1, x_2, \ldots, x_{J-1}, x_J)'$. Replacing the scalar x_0 by the last element of the vector, namely x_J, yields the next vector $(x_{J+1}, x_{J+2}, \ldots, x_{2J-1}, x_{2J})'$, etc. In this way the pseudorandom numbers are generated in exactly the same order as they would have been produced in scalar mode.

At the end of the simulation run we should store the last pseudorandom number, so that the simulation experiment might be continued later on, or a new (unrelated) simulation experiment can start at a seed different from the default x_0. Also see De Matteis and Pagnutti (1988, p. 602). We shall return to this generator, after we have discussed a closely related generator.

4.2. Vector of J successive numbers

Suppose we have available of vector of J successive pseudorandom numbers

$$\underset{\sim}{x} = (x_0, x_1, x_2, \ldots, x_{J-2}, x_{J-1})'. \tag{4.3}$$

This vector is multiplied by the scalar multiplier (a^J) mod m. This multiplication gives a new vector identical to the new vector obtained by equation (4.2). At the end of a simulation we should store the vector of the last J numbers.

There is a computational problem in both approaches: overflow occurs when computing high powers of the multiplier a. (Overflow in classic generators is also discussed in Park and Miller, 1988, p. 1195.) That problem, however, can be solved if we use 'controlled integer overflow'; see Law and Kelton (1982, pp. 219-232). We also must know that the CYBER 205 uses the 'two 's complement' representation of negative integers. The Appendix gives the computer program for the generator based on equation (4.3) (which will turn out to be the most efficient generator).

4.3. Comparison of four generators

Table 3 compares the computer efficiency of several generators, using the CYBER 205. This computer can use FORTRAN 200 (a superset of FORTRAN 77) that allows vector and scalar programming; see CDC (1986): Generator #1 is RANF, a scalar subroutine that uses the multiplicative generator with $m = 2^{47}$ and a = 84000335758957 (or in hexadecimal notation, a = 00004C65DA2C866D); see CDC (1986, pp. 10-29). The CYBER 205 uses words of 64 bits; 48 bits are used to represent integers, including one sign bit; hence $m = 2^{47}$. Generator #2 is VRANF, a standard vectorized subroutine that uses the same m and a as RANF does; see CDC (1986, pp. 11-1). Generator #3 uses the vector of multipliers of equation (4.2); generator #4 uses the vector of J preceding numbers x_j plus the multiplier a^J; see equation (4.3); these two generators use the same multiplier and modulus as RANF does. We can implement the last two generators not only in vector mode but also in scalar mode (of course RANF is in scalar mode, and VRANF is in vector mode). The measurements in Table 3 do not include storing the last vector or scalar to continue simulation at the last pseudorandom number.

Table 3: Computer time in microseconds on CYBER 205.

Type of generator	vector length J			
	5	500	50,000	65,535
#1 RANF	0.014	0.520	51.553	67.465
#2 VRANF	0.021	0.208	19.507	25.652
#3 J Multipliers	0.013	0.079	7.713	9.923
scalar mode	0.026	1.572	157.763	206.843
#4 J numbers & a^J	0.013	0.079	7.425	9.631
scalar mode	0.024	1.561	157.098	206.083

Our results for RANF and VRANF deviate substantially from those published by An Mey (1983): he found that VRANF is always slower than RANF and he found CPU times a factor 1,000 higher! Generator #4 is slightly fester than generator #3 because the latter generator has to store and fetch the last element of the vector of numbers x_j. Moreover generator #3 needs two vectors, namely one vector for the multipliers a_j and one vector for the numbers x_j. So we recommend generator #4.

5. Conclusions

Multiplicative generators show very strong 'long range' correlations. Splitting up the cycle of such a generator into 2^k parts results in pseudorandom numbers that lie on no more than 2^{k-1} parallel lines, if $k \geq 3$. On supercomputers, pseudorandom numbers could be generated by partitioning into 2^5 parts and using only the first two parts. Two better approaches require the computation, once and for all, of J multipliers a_j =

a^j mod m, or the computation of the multiplier a^J mod m and initializing a vector with J successive numbers. These two approaches are superior to the standard subroutines on the CYBER 205, namely RANF and VRANF.

Acknowledgement

We thank Bert Bettonvil for his useful comments on earlier versions of this paper.

Appendix 1: The FORTRAN 200 program for generator £4.

```
      PROGRAM VARIANT4
      IMPLICIT REAL (U-Z), INTEGER (A-T)
      PARAMETER (N1=5,N4=65535,K=1)
      PARAMETER (A1=37772072706109)
      INTEGER MVAST
      BIT BVAST
      DESCRIPTOR MVAST, BVAST
      DIMENSION T(N4), S1(N1)
      DIMENSION X1(N1)
      DATA MINT / X'0000800000000000' /
      CALL RANSET(K)
      DO 5 I=1,N4
       U=RANF()
       CALL RANGET(T(I))
    5 CONTINUE
C     ! N=5
C     ! SCALAR
      S1(1;N1)=T(1;N1)
      ZPU1=SECOND()
      DO 10 I=1,N1
       S1(I)=A1*S1(I)
       IF (S1(I).LT.0) S1(I)=S1(I)-MINT
       X1(I)=S1(I)/MINT
   10 CONTINUE
      ZPU2=SECOND()
      U1=ZPU2-ZPU1
C     ! VECTOR
      ASSIGN MVAST,.DYN.N1
      ASSIGN BVAST,.DYN.N1
      S1(1;N1)=T(1;N1)
      ZPU1=SECOND()
      S1(1;N1)=A1*S1(1;N1)
      BVAST=S1(1;N1).LT.0
      MVAST=S1(1;N1)-MINT
```

```
S1(1;N1)=Q8VCTRL(MVAST,BVAST;S1(1;N1))
X1(1;N1)=S1(1;N1)/MINT
ZPU2=SECOND()
Z1=ZPU2-ZPU1
FREE
PRINT *, 'BEGIN:   GEVEKTORISEERD   SCALAR'
PRINT *, 'N=     5  ',Z1,'  ',U1
END
```

References

An Mey, D., Erste Erfahrungen bei der Vektorisiering numerischer Verfahren. (First experiences when vectorizing numerical procedures.) Computer Center, Technical University, Aachen (Germany), July 1983.

Bratley, P., B.L. Fox and L.E. Schrage, A GUIDE TO SIMULATION, Springer-Verlag, New York, 1983.

CDC, Fortran 200 Version 1 reference manual. Publication no. 60480200, Control Data Corporation, Sunnyvale, California 94088-3492, December 1986.

Churchill, G., Marketing Research. 3rd ed., Dryden Press, Chicago, 1983.

De Matteis, A. and S. Pagnutti, Parallelization of random number generators and long-range correlations. NUMERISCHE MATHEMATIK, 53, 1988, pp. 595-608.

Fishman, G.S., PRINCIPLES OF DISCRETE EVENT SIMULATION. Wiley-Interscience, New York, 1978.

Kleijnen, J.P.C., Pseudorandom number generation on supercomputers, Catholic University Brabant (Katholieke Universiteit Brabant), May 1989.

Law, A. and W. Kelton, Simulation modelling and analysis. McGraw-Hill, New York, 1982.

Levine, R.D., Supercomputers. SCIENTIFIC AMERICAN, January 1982, pp. 112-125.

Park, S.K. and K.W. Miller, Random number generators: good ones are hard to find. COMMUNICATIONS ACM, 31, no. 10, Oct. 1988, pp. 1192-1201.

Ripley, B.D., STOCHASTIC SIMULATION. John Wiley & Sons, New York, 1987.

SARA, CYBER 205 USER'S GUIDE; PART. 3, OPTIMIZATION OF FORTRAN PROGRAMS. SARA (Stichting Academisch Rekencentrum Amsterdam, Foundation Academic Computer Center Amsterdam), Amsterdam, November 1984.

Figure 1: Plot of all successive pairs x_{2j}, x_{1+2j} with $j = 0, 1, \ldots, h/2-1$ for a multiplicative generator with $m = 2^6$ and $a = 5$.

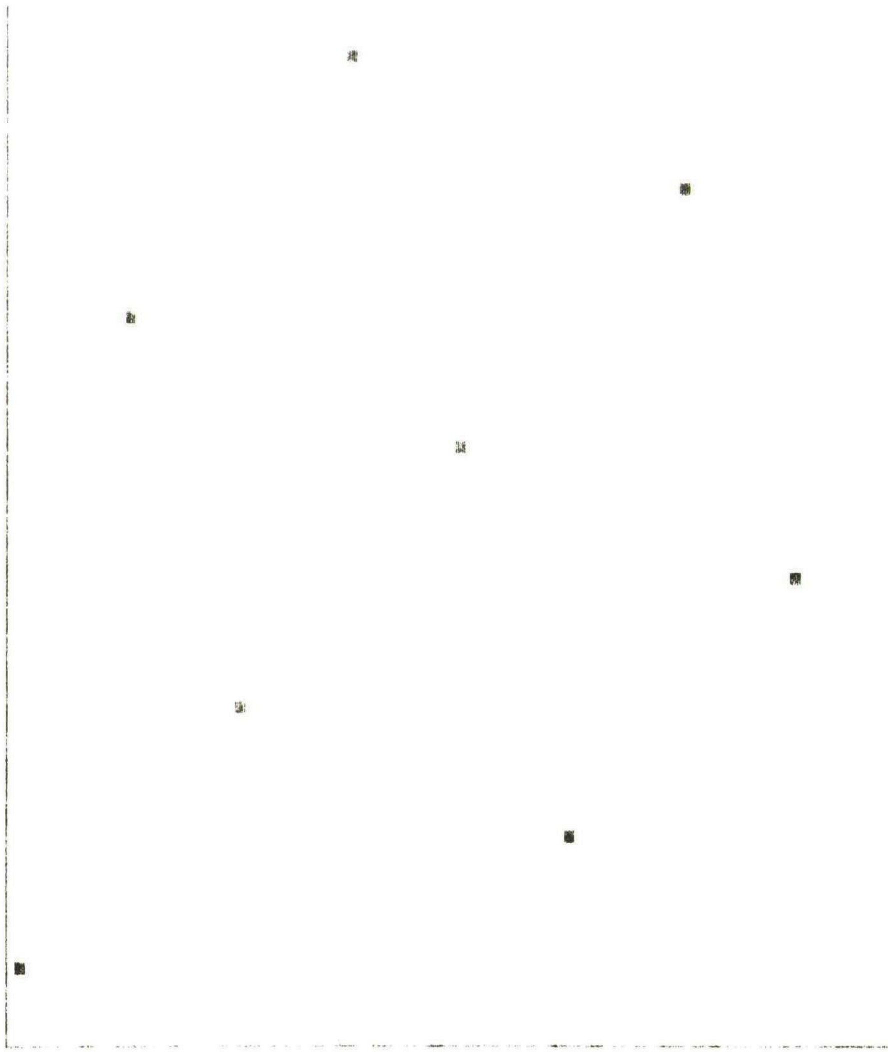


Figure 2. Pairs across two parts $(x_j, x_{j+h/2})$ with j = 0, ..., h/2-1 for multiplicative generator with $m = 2^6$ and a = 5.

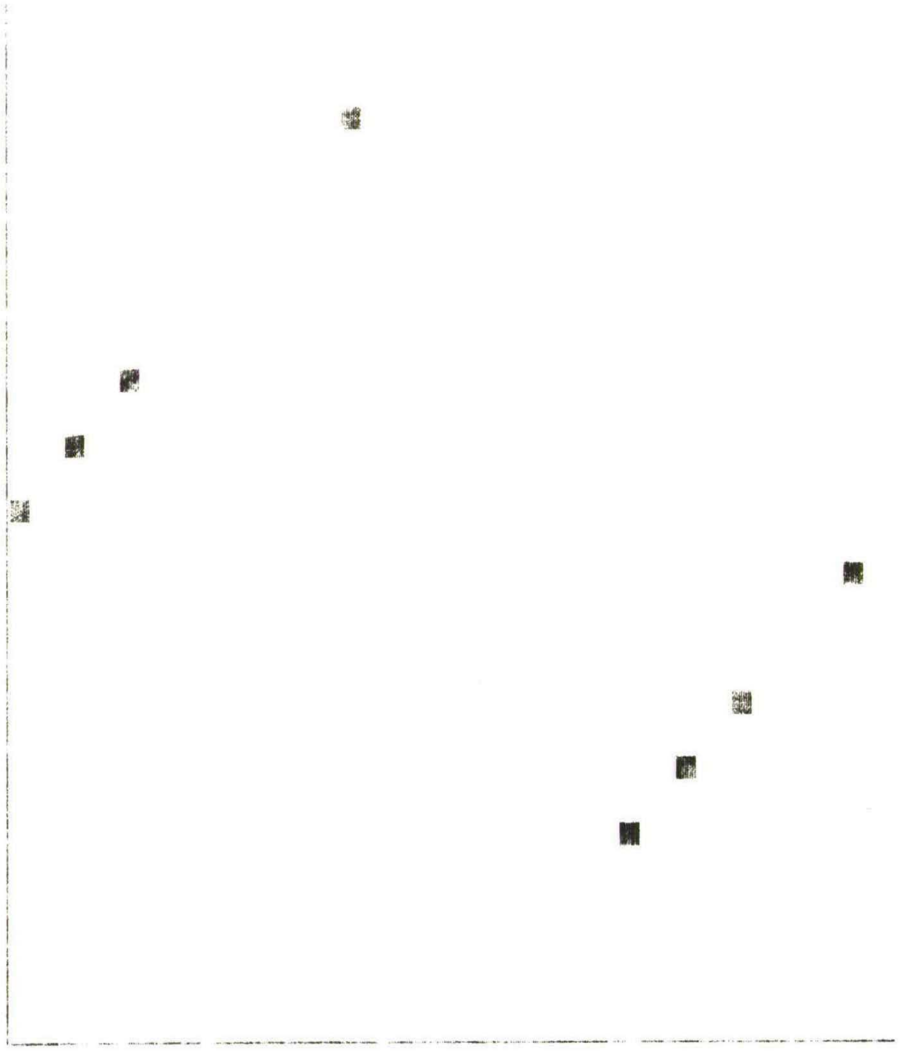


Figure 3. Pairs across two parts.

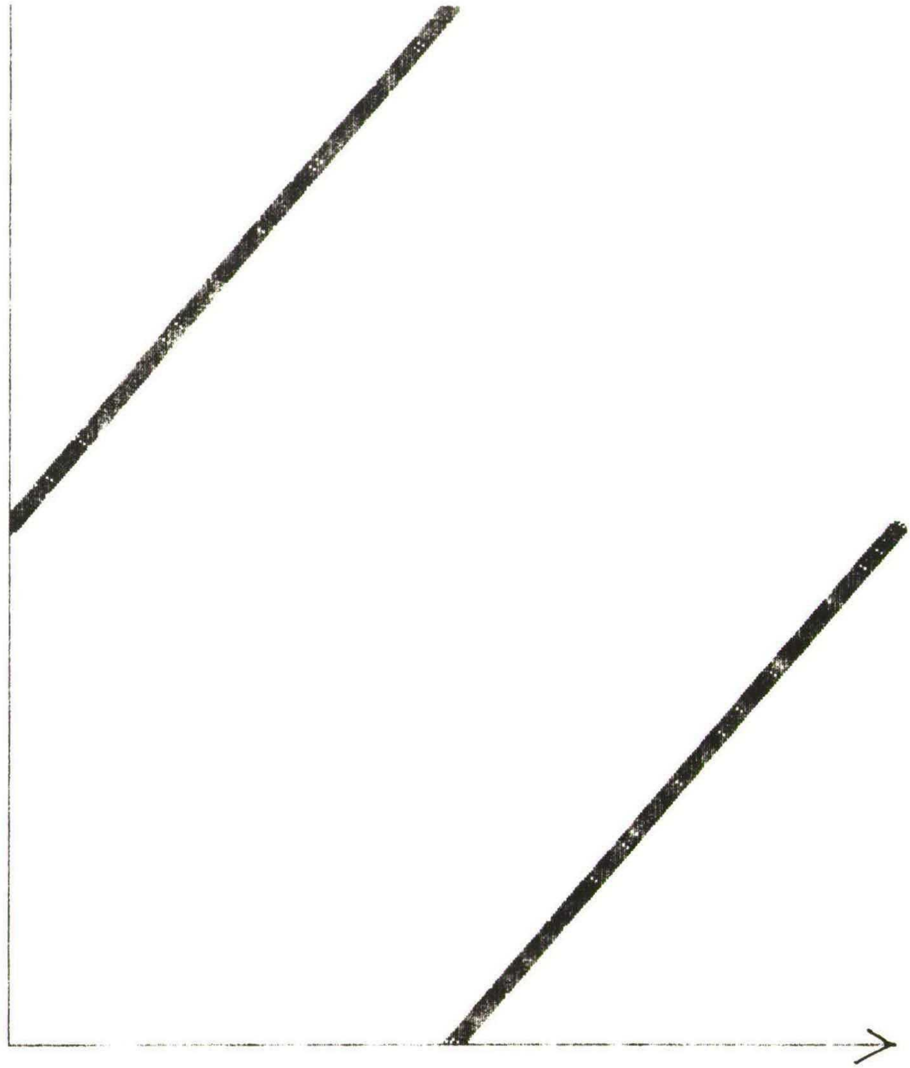


Figure 4. Pairs across first two parts when splitting up into four equal parts.

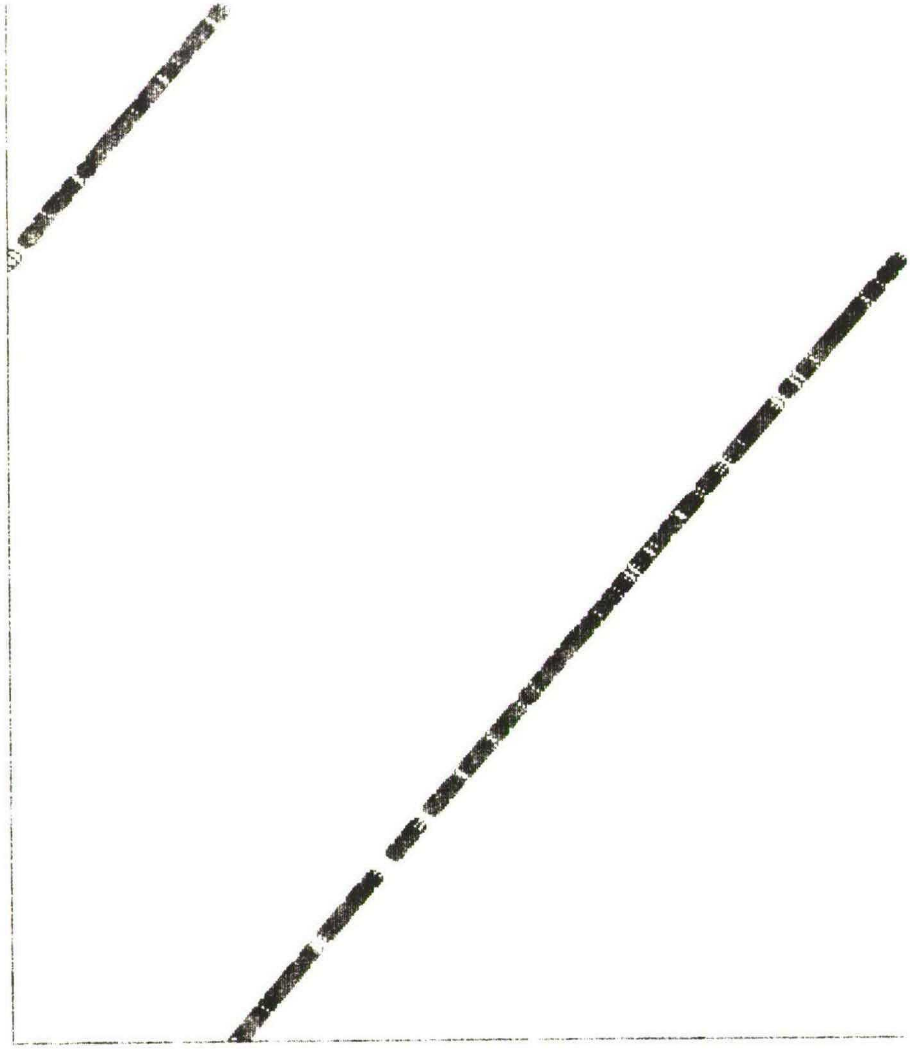


Figure 5. Pairs across first two parts for 2^3 parts.

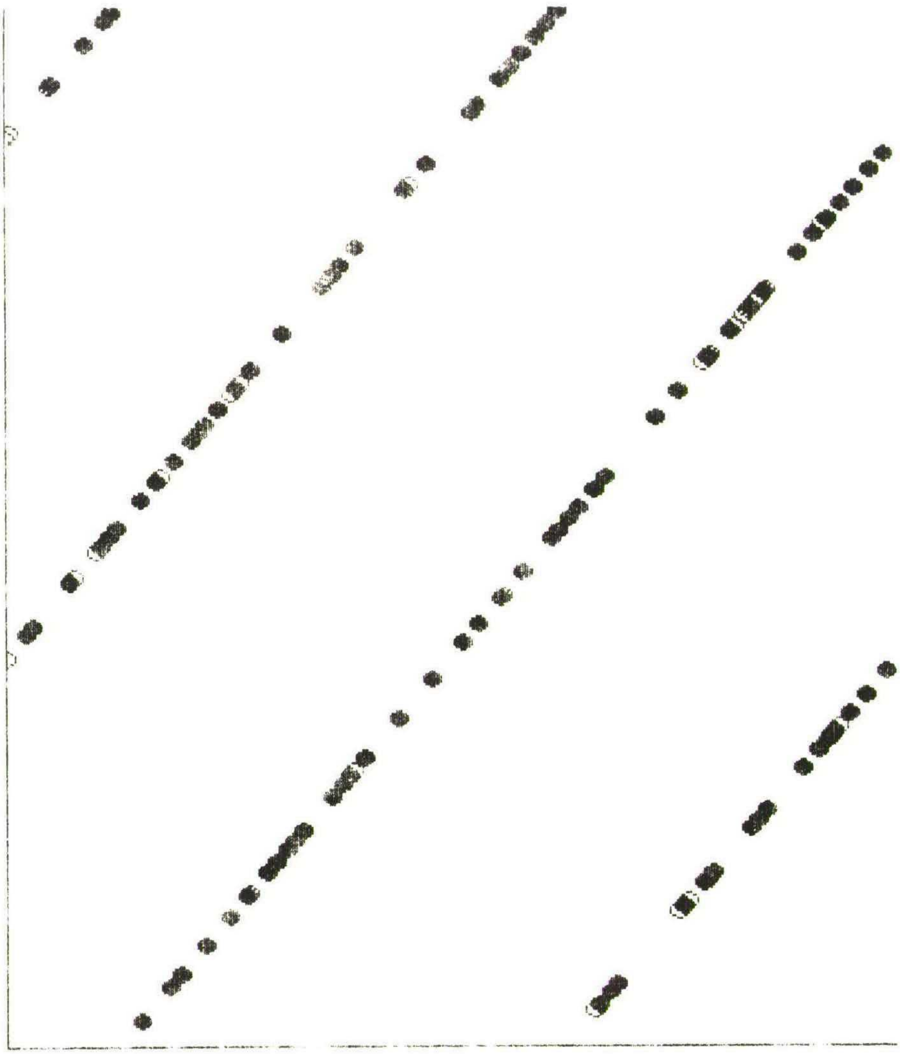


Figure 6. Pairs across two parts for 2^4 parts.

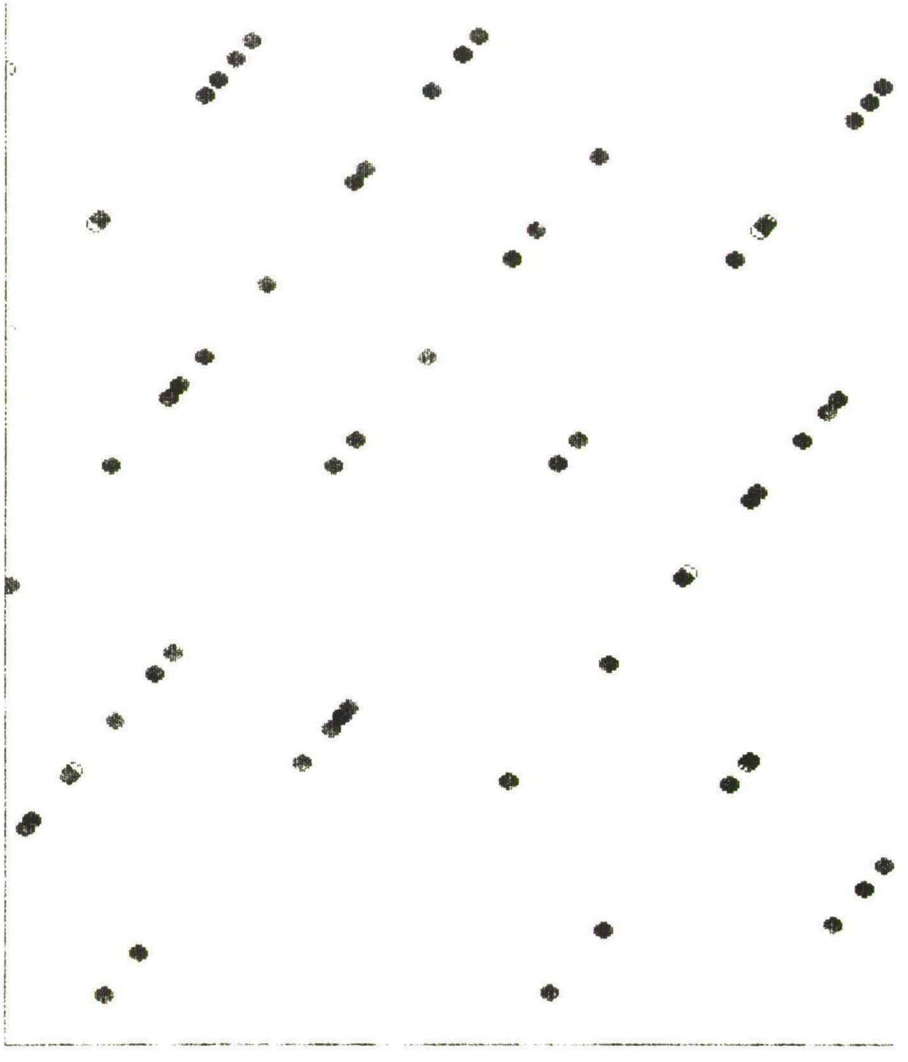
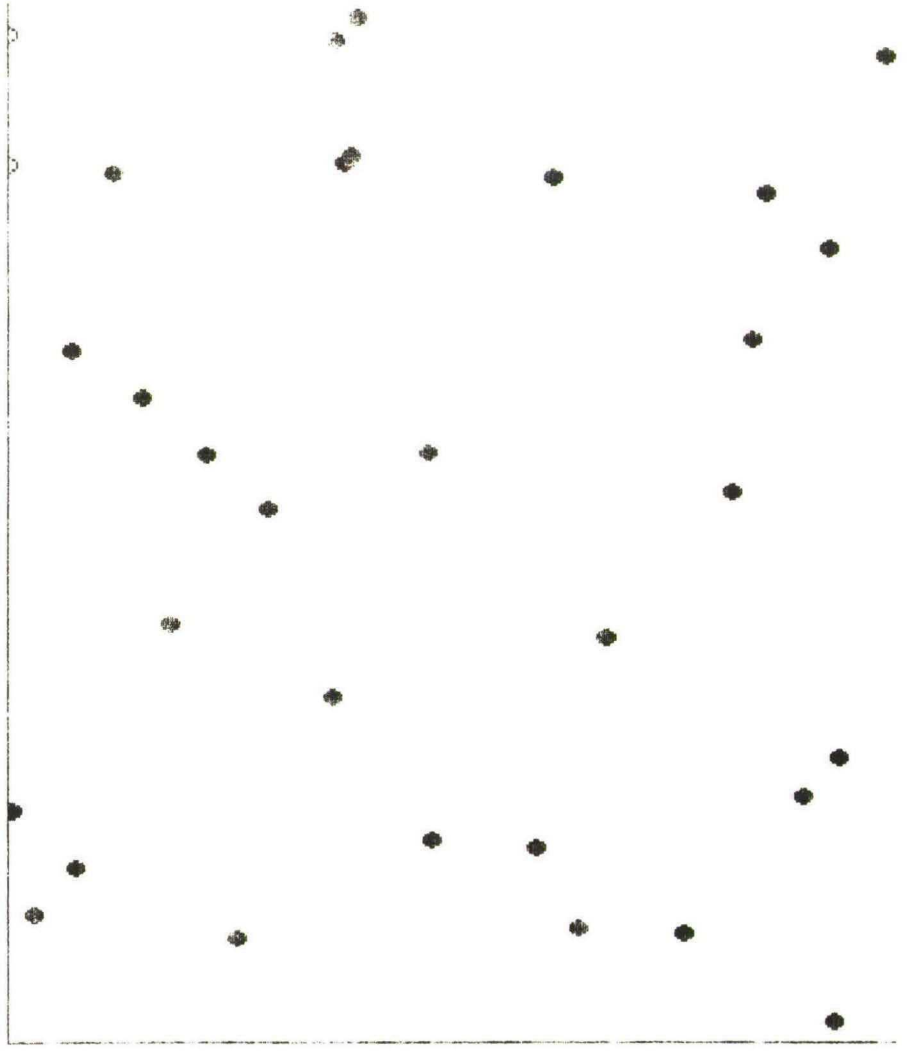


Figure 7. Pairs across two parts for 2^5 parts.



IN 1988 REEDS VERSCHENEN

297 Bert Bettonvil
Factor screening by sequential bifurcation

298 Robert P. Gilles
On perfect competition in an economy with a coalitional structure

299 Willem Selen, Ruud M. Heuts
Capacitated Lot-Size Production Planning in Process Industry

300 J. Kriens, J.Th. van Lieshout
Notes on the Markowitz portfolio selection method

301 Bert Bettonvil, Jack P.C. Kleijnen
Measurement scales and resolution IV designs: a note

302 Theo Nijman, Marno Verbeek
Estimation of time dependent parameters in lineair models using cross sections, panels or both

303 Raymond H.J.M. Gradus
A differential game between government and firms: a non-cooperative approach

304 Leo W.G. Strijbosch, Ronald J.M.M. Does
Comparison of bias-reducing methods for estimating the parameter in dilution series

305 Drs. W.J. Reijnders, Drs. W.F. Verstappen
Strategische bespiegelingen betreffende het Nederlandse kwaliteitsconcept

306 J.P.C. Kleijnen, J. Kriens, H. Timmermans and H. Van den Wildenberg
Regression sampling in statistical auditing

307 Isolde Woittiez, Arie Kapteyn
A Model of Job Choice, Labour Supply and Wages

308 Jack P.C. Kleijnen
Simulation and optimization in production planning: A case study

309 Robert P. Gilles and Pieter H.M. Ruys
Relational constraints in coalition formation

310 Drs. H. Leo Theuns
Determinanten van de vraag naar vakantiereizen: een verkenning van materiële en immateriële factoren

311 Peter M. Kort
Dynamic Firm Behaviour within an Uncertain Environment

312 J.P.C. Blanc
A numerical approach to cyclic-service queueing models

313 Drs. N.J. de Beer, Drs. A.M. van Nunen, Drs. M.O. Nijkamp
Does Morkmon Matter?

314 Th. van de Klundert
Wage differentials and employment in a two-sector model with a dual labour market

315 Aart de Zeeuw, Fons Groot, Cees Withagen
On Credible Optimal Tax Rate Policies

316 Christian B. Mulder
Wage moderating effects of corporatism
Decentralized versus centralized wage setting in a union, firm, government context

317 Jörg Glombowski, Michael Krüger
A short-period Goodwin growth cycle

318 Theo Nijman, Marno Verbeek, Arthur van Soest
The optimal design of rotating panels in a simple analysis of variance model

319 Drs. S.V. Hannema, Drs. P.A.M. Versteijne
De toepassing en toekomst van public private partnership's bij de grote en middelgrote Nederlandse gemeenten

320 Th. van de Klundert
Wage Rigidity, Capital Accumulation and Unemployment in a Small Open Economy

321 M.H.C. Paardekooper
An upper and a lower bound for the distance of a manifold to a nearby point

322 Th. ten Raa, F. van der Ploeg
A statistical approach to the problem of negatives in input-output analysis

323 P. Kooreman
Household Labor Force Participation as a Cooperative Game; an Empirical Model

324 A.B.T.M. van Schaik
Persistent Unemployment and Long Run Growth

325 Dr. F.W.M. Boekema, Drs. L.A.G. Oerlemans
De lokale produktiestructuur doorgelicht.
Bedrijfstakverkenningen ten behoeve van regionaal-economisch onderzoek

326 J.P.C. Kleijnen, J. Kriens, M.C.H.M. Lafleur, J.H.F. Pardoel
Sampling for quality inspection and correction: AOQL performance criteria

327 Theo E. Nijman, Mark F.J. Steel
Exclusion restrictions in instrumental variables equations

328 B.B. van der Genugten
Estimation in linear regression under the presence of heteroskedasticity of a completely unknown form

329 Raymond H.J.M. Gradus
The employment policy of government: to create jobs or to let them create?

330 Hans Kremers, Dolf Talman
Solving the nonlinear complementarity problem with lower and upper bounds

331 Antoon van den Elzen
Interpretation and generalization of the Lemke-Howson algorithm

332 Jack P.C. Kleijnen
Analyzing simulation experiments with common random numbers, part II: Rao's approach

333 Jacek Osiewalski
Posterior and Predictive Densities for Nonlinear Regression. A Partly Linear Model Case

334 A.H. van den Elzen, A.J.J. Talman
A procedure for finding Nash equilibria in bi-matrix games

335 Arthur van Soest
Minimum wage rates and unemployment in The Netherlands

336 Arthur van Soest, Peter Kooreman, Arie Kapteyn
Coherent specification of demand systems with corner solutions and endogenous regimes

337 Dr. F.W.M. Boekema, Drs. L.A.G. Oerlemans
De lokale produktiestruktuur doorgelicht II. Bedrijfstakverkenningen ten behoeve van regionaal-economisch onderzoek. De zeescheepsnieuwbouwindustrie

338 Gerard J. van den Berg
Search behaviour, transitions to nonparticipation and the duration of unemployment

339 W.J.H. Groenendaal and J.W.A. Vingerhoets
The new cocoa-agreement analysed

340 Drs. F.G. van den Heuvel, Drs. M.P.H. de Vor
Kwantificering van ombuigen en bezuinigen op collectieve uitgaven 1977-1990

341 Pieter J.F.G. Meulendijks
An exercise in welfare economics (III)

357 R.M. Heuts, M. Bronckers
A modified coordinated reorder procedure under aggregate investment and service constraints using optimal policy surfaces

358 B.B. van der Genugten
Linear time-invariant filters of infinite order for non-stationary processes

359 J.C. Engwerda
LQ-problem: the discrete-time time-varying case

360 Shan-Hwei Nienhuys-Cheng
Constraints in binary semantical networks

361 A.B.T.M. van Schaik
Interregional Propagation of Inflationary Shocks

362 F.C. Drost
How to define UMVU

363 Rommert J. Casimir
Infogame users manual
Rev 1.2 December 1988

364 M.H.C. Paardekooper
A quadratically convergent parallel Jacobi-process for diagonal dominant matrices with nondistinct eigenvalues

365 Robert P. Gilles, Pieter H.M. Ruys
Characterization of Economic Agents in Arbitrary Communication Structures

366 Harry H. Tigelaar
Informative sampling in a multivariate linear system disturbed by moving average noise

367 Jörg Glombowski
Cyclical interactions of politics and economics in an abstract capitalist economy

IN 1989 REEDS VERSCHENEN

368 Ed Nijssen, Will Reijnders
"Macht als strategisch en tactisch marketinginstrument binnen de distributieketen"

369 Raymond Gradus
Optimal dynamic taxation with respect to firms

370 Theo Nijman
The optimal choice of controls and pre-experimental observations

371 Robert P. Gilles, Pieter H.M. Ruys
Relational constraints in coalition formation

372 F.A. van der Duyn Schouten, S.G. Vanneste
Analysis and computation of (n,N)-strategies for maintenance of a two-component system

373 Drs. R. Hamers, Drs. P. Verstappen
Het company ranking model: a means for evaluating the competition

374 Rommert J. Casimir
Infogame Final Report

375 Christian B. Mulder
Efficient and inefficient institutional arrangements between governments and trade unions; an explanation of high unemployment, corporatism and union bashing

376 Marno Verbeek
On the estimation of a fixed effects model with selective non-response

377 J. Engwerda
Admissible target paths in economic models

378 Jack P.C. Kleijnen and Nabil Adams
Pseudorandom number generation on supercomputers

379 J.P.C. Blanc
The power-series algorithm applied to the shortest-queue model

380 Prof. Dr. Robert Bannink
Management's information needs and the definition of costs, with special regard to the cost of interest

381 Bert Bettonvil
Sequential bifurcation: the design of a factor screening method

382 Bert Bettonvil
Sequential bifurcation for observations with random errors

383 Harold Houba and Hans Kremers
Correction of the material balance equation in dynamic input-output models

384 T.M. Doup, A.H. van den Elzen, A.J.J. Talman
Homotopy interpretation of price adjustment processes

385 Drs. R.T. Frambach, Prof. Dr. W.H.J. de Freytas
Technologische ontwikkeling en marketing. Een oriënterende beschouwing

386 A.L.P.M. Hendrikx, R.M.J. Heuts, L.G. Hoving
Comparison of automatic monitoring systems in automatic forecasting

387 Drs. J.G.L.M. Willems
Enkele opmerkingen over het inversificerend gedrag van multinationale ondernemingen