1968-06

Methods for Computing the Greatest Common Divisor and Applications in Mathematical Programming.

MacGregor, Harry Gregor, Jr.

Monterey, California. Naval Postgraduate School

METHODS FOR COMPUTING THE GREATEST COMMON DIVISOR AND APPLICATIONS IN MATHEMATICAL PROGRAMMING

HARRY GREGOR MAC GREGOR, JR.
and
KENT ALLEN MODINE

METHODS FOR COMPUTING THE GREATEST COMMON DIVISOR

AND APPLICATIONS IN MATHEMATICAL PROGRAMMING

by

Harry Gregor MacGregor, Jr.
Major, United States Army
B.S.C.E., Virginia Military Institute, 1959

and

Kent Allen Modine
Captain, United States Army
B.S.C.E., Virginia Military Institute, 1961

[redacted]

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 1968

ABSTRACT

Several methods are presented for determining the greatest common divisor of a set of positive integers by solving the integer program: find the integers x_i that minimize $Z = \sum_{i=1}^{n} a_i x_i$ subject to $Z \geq 1$. The methods are programmed for use on a computer and compared with the Euclidean algorithm. Computational results and applications are given.

TABLE OF CONTENTS

LIST OF TABLES

1. Introduction

The Euclidean algorithm provides a method for computing the greatest common divisor (GCD) of a set of positive integers $a_1, a_2, \ldots, a_n$. The problem can also be solved as an integer program: find integers $x_1, x_2, \ldots, x_n$ that minimize

$$z = \sum_{i=1}^{n} a_i x_i \tag{1}$$

where $z \geq 1$.

Proof that the optimal value of z is the GCD of the a_i is contained in the following theorems.

Theorem 1. The minimum positive integer in the set $L = \sum_{i=1}^{n} a_i x_i$, where the a_i are given positive integers and the x_i are all possible integers, is the GCD of the set.

There is a finite number of integers between zero and any positive integer. The set L contains at least one positive integer, therefore the set has a minimum positive integer. Denote the minimum positive integer by

$$M = \sum_{i=1}^{n} a_i x'_i. \tag{2}$$

By Euclid's theorem [7] for any integer S there exist integers p and q such that

$$S = pM + q, \qquad 0 \leq q < M. \tag{3}$$

For S in L we also have

$$S = \sum_{i=1}^{n} a_i x''_i. \tag{4}$$

From equations (2), (3), and (4) we obtain:

$$\sum_{i=1}^{n} a_i x_i'' = p \sum_{i=1}^{n} a_i x'_i + q$$

or

$$\sum_{i=1}^{n} a_i (x_i'' - p x_i') = q.$$

Since $(x_i'' - px_i')$, for all i, are integers, q is an integer contained in L. Since q is less than M and M is the minimum positive integer in L, q must equal zero. From equation (3) we see that M divides S and thus it divides every member of L and is a common divisor of L. Since M is in L, no integer greater than M is a common divisor of L. Therefore M is the GCD of L.

Theorem 2. The GCD of the a_i is the minimum positive integer in the set $L = \sum_{i=1}^{n} a_i x_i$.

Each integer a_i is in L and may be determined when $x_i = 1$ and $x_k = 0$ for $k \neq i$. Therefore, from theorem 1, M is a common divisor of all a_i. Since $M = \sum_{i=1}^{n} a_i x_i'$ any common divisor of the a_i divides M. Therefore the GCD cannot exceed M and thus M is the GCD of the a_i.

Theorem 3. The GCD of the set $L = \sum_{i=1}^{n} a_i x_i$ is unique.

Let M_1 and M_2 be greatest common divisors of L. Since M_1 and M_2 are in L, M_1 must divide M_2 and M_2 must divide M_1. Consequently, $M_1 \leq M_2$ and $M_2 \leq M_1$. Therefore $M_1 = M_2$.

In many applications it is desirable to find the elements x_i; the Euclidean algorithm does this in a somewhat tedious fashion. Blankinship [2] provides a matrix method that duplicates the Euclidean algorithm and produces the x_i. But his procedure requires the storage of an n by n + 1 matrix and if n is large the method runs into storage problems.

We will present an algorithm for the solution of the integer program (1) that does not have the storage problem of [2] and also achieves computer solutions more rapidly. In addition, non-unique solutions can be produced easily, which the Euclidean algorithm and [2] can not readily achieve.

2. Solution of the Integer Program

The program (1) is equivalent to the problem: find integers $x_1, x_2, \ldots, x_n$ that

$$\text{minimize } x_{n+1}$$

$$\text{subject to } \sum_{i=1}^{n} a_i x_i - x_{n+1} = 1 \tag{5}$$

and $x_{n+1} \geq 0$. The solution to (1) is then $z = 1 + x_{n+1}$ for optimal x_{n+1}. The result may be obtained by using (5) directly until the integer program is solved by inspection. This procedure is contained in algorithm 1 as follows:

1. Set $m = 1$ and define $a_{1i} = a_i$ for all i. Go to 2.
2. (a) Find $a_{mr} = \min a_{mi} > 0$. Set $R(m) = r$ and $D_m = \text{GCD}(a_{mr}, D_{m-1})$ for $m \neq 1$ or $D_m = a_{mr}$ for $m = 1$. If $D_m = 1$ go to 3. Otherwise go to 2(b).

   (b) Calculate $a_{m+1,i} = a_{mi} \pmod{D_m}$. If all $a_{m+1,i} = 0$ go to 3. Otherwise set $m = m+1$ and go to 2(a).
3. The problem is solved; $x_{n+1} = -1 + D_m$ with $z = D_m$. To find the x_i set $M = m$ and go to 4.
4. (a) If $m = 1$ go to 4(c). Otherwise go to 4(b).

   (b) Set $i = R(m)$. If $M = m$, calculate x_i from $a_{mi} x_i = z \pmod{D_{m-1}}$. Otherwise calculate x_i from $a_{mi} x_i = z - \sum_{j=m+1}^{M} a_{mk} x_k \pmod{D_{m-1}}$, where $k = R(j)$. In either case set $m = m-1$ and go to 4(a).

(c) Set $i = R(1)$. If $M = 1$, calculate x_i from $a_{1i}x_i = z$. Otherwise calculate x_i from $a_{1i}x_i = z - \sum_{j=2}^{M} a_{1k}x_k$, where $k = R(j)$. Stop.

This completes the algorithm. As an example we take the problem in [2]; $a_1 = 99$, $a_2 = 77$, $a_3 = 63$. We list successively

99	77	63	:	$D_1 = 63$,	$R(1) = 3$
36	14	0	:	$D_2 = 7$,	$R(2) = 2$
1	0	0	:	$D_3 = 1$,	$R(3) = 1$

Thus $z = 1$. Backtracking we have $x_1 = 1 \pmod 7$. Using $x_1 = 1$, we have $14x_2 = -35 \pmod{63}$; using $x_2 = 2$, we have $63x_3 = -252$, which results in $x_3 = -4$. Other x_i may be found readily from the multiple solutions, to the congruences. All possible solutions are given by the solutions to the set of equations $x_1 = 1 \pmod 7$, $x_2 = 2 \pmod 9$, $x_3 = 7 \pmod{11}$, and $99x_1 + 77x_2 + 63x_3 = 1$. A justification of the algorithm follows:

1) Since all the x_i are integer we can obtain the congruence

$$\sum_{i \neq r} a_{2i}x_i - x_{n+1} = 1 \pmod{D_1} \text{ from (5).} \qquad (6)$$

2) We next consider the program min x_{n+1} subject to (6). We need not consider (5) as a constraint since x_r is not sign restricted. We maintain (5) to calculate x_r.

3) At any stage in the algorithm we develop a congruence

$$\Sigma a_{mi}x_i - x_{n+1} = 1 \pmod{D_{m-1}}. \qquad (7)$$

4) We then consider the program min x_{n+1} subject to (7). We need not consider any previous congruence since the x_i, $i = 1,\ldots,n$ are not sign restricted.

5) We calculate $D_m = \text{GCD}(a_{mr}, D_{m-1})$; if $D_m = 1$, we can solve (7), with $x_{n+1} = 0$, $x_i = 0$ $(i \neq r)$ and x_r given by $a_{mr} x_r = 1 \pmod{D_{m-1}}$. Thus we have $z = 1$. If $D_m \neq 1$, then D_m must divide $1 + x_{n+1} - \Sigma a_{mi} x_i$. We then produce another congruence with m replaced by m+1 in (7).

6) If all $a_{m+1,i} = 0$ then $-x_{n+1} = 1 \pmod{D_m}$, which results in $z = D_m$.

In Algorithm 1 we require the GCD of pairs of numbers. We can use either the usual Euclidean algorithm or a variant of Algorithm 1 by maintaining the $a_{m,r}$ value for $a_{m+1,r}$ (instead of being zero). In this way D_m will be listed.

As seen in the example the congruences simplify; e.g., in (7), D_m divides D_{m-1} as well as the numbers in the congruence. In computer calculation we have obtained the solution to the reduced congruence by enumeration.

The storage required for Algorithm 1 is essentially the product of n and the number of times required to perform step 2 of the algorithm. The storage problem may be reduced further by stopping the algorithm after completing step 2 and solving the program: minimize x_{n+1} subject to (7). Taking $D = D_{m-1}$ and $b_i = a_{mi} > 0$, we define

$$F(x) = \min \left(x_{n+1} \middle| \Sigma b_i x_i - x_{n+1} = x \pmod{D}\right), \quad (8)$$

which is equivalent to the dynamic programming recursion

$$F(x) = \min\,(1+F(x+1),\ \min\,(F(x-b_i),\ F(x+b_i))) \qquad (9)$$

$$F(0) = 0.$$

The arguments of F are taken modulo D. For a similar recursion, see [3]. The recursion in (9) is solved in a manner similar to that given in [4] by

Algorithm 2:

1. Set $F(x) = k \geq D$ for $x = 1,2,\ldots,D-1$. Go to 2.
2. (a) Set $x = 1$ and $t = 0$. Go to 2(b).

   (b) Calculate

   $$G(x) = \min\,(1+F(x+1),\ \min\,(F(x-b_i),\ F(x+b_i)))$$

   Set $R(x) = n+1$ if the minimum occurs for the $1+F(x+1)$ term.

   Set $R(x) = i$ if the minimum occurs for $F(x-b_i)$.

   Set $R(x) = -i$ if the minimum occurs for $F(x+b_i)$.

   (c) If $G(x) = F(x)$ go to 3. Otherwise set $F(x) = G(x)$ and $t = 1$ and go to 3.
3. Several cases are possible:

   (a) if $x = D-1$ and $t = 0$, go to 4.

   (b) if $x = D-1$ and $t = 1$, go to 2(a).

   (c) if $x < D-1$, set $x = x+1$ and go to 2(b).
4. Solution is achieved with $z = 1 + F(1)$. The values of the x_i are found as follows:

   (a) Set all $x_i = 0$. Set $x = 1$ and go to 4(b).

   (b) If $R(x) = i > 0$ for $i \neq n+1$ set $x_i = x_i + 1$, $s = -a_i$ and go to 4(c). If $R(x) = n+1$ set $s = 1$ and go to 4(c). Otherwise $R(x) = i < 0$; set $x_i = x_i - 1$, $s = a_i$ and go to 4(c).

(c) Set $x = x+s \pmod D$. If $x = 0$ go to 5. Otherwise go to 4(b).

5. The final x_i values are the desired ones. Stop.

This completes the algorithm. Alternate values of the x_i may be obtained by taking ties into account in the recursion. Algorithm 2 completes the recursion in (9) in a rapid manner due to the profusion of zeroes that arise for the various $F(x)$. The recursion is completed in a finite number of steps as shown in [4].

3. Computer Programming of Algorithms

To determine the best algorithm for computer use, we programmed the Blankinship method [2] and several variations of the algorithms described in section 2. The four methods programmed are as follows;

(i) Blankinship method. The algorithm was programmed as outlined in [2] to calculate the GCD and the x_i as given in Appendix I. We also programmed a modified version of this algorithm which calculates only the GCD. This program is given in Appendix II.

(ii) Algorithm 1 method. Algorithm 1 was programmed as given in Appendix III.

(iii) Algorithm 2 method. Algorithm 2 was programmed as given in Appendix IV. A modified version of this algorithm was programmed to calculate only the GCD and is given in Appendix V.

(iv) Combination method. This method combines algorithms 1 and 2. If D is less than or equal to k (determination of k to be discussed in section 4) then we use algorithm 2. If D is greater than k use algorithm 1 until D is less than or equal to k then use algorithm 2. After completion of algorithm 2, the final x_i values are calculated using step 4c of algorithm 1. The program for this method is given in Appendix VI. This method was also programmed to calculate only the GCD as given in Appendix VII.

4. Computational Results

We have programmed the seven methods in Fortran IV and have measured their execution times on a series of test problems run on an IBM 360/67 computer. The test problems were designed to calculate the GCD of a given number of integers, N, over a specified range, R, and a controlled GCD. Since a group of random numbers usually have a GCD of one, we controlled the GCD by generating numbers as multiples of the desired GCD. The ten combinations of R and N for each of three greatest common divisors as shown in Table 1 give 30 test conditions. Three sets of integers were used for each of the test conditions which resulted in 90 test problems. The 90 problems were used to test each of the seven methods.

TABLE I. Test Conditions

R \ N	10	50	100	250
1-1000	x	x	x	x
1-500	x	x	x	
500-1000	x	x	x	

An x indicates the combination was used. Each was used with GCD of 1, 2, and 3.

We used as a basis of comparison of the efficiency of these methods, the computer storage requirement and execution time of the problem. Computer storage requirement was not a significant factor except for the Blankinship method. The requirement for more than N^2 words of storage is a serious limitation of the Blankinship method for large N.

The average execution times for the test conditions are given in Table II. Examination of these results indicates that the Algorithm I method is the superior method. It may be noted that the Blankinship method is competative with a small number of integers and a GCD of one. The Combination method is competative when all the integers are large. Therefore it may be the preferred method when computing the GCD of large numbers or when computer storage is critical. Our computational experience indicates that the best results are obtained from the Combination method when k is approximately equal to 1.5 $(N)^{\frac{1}{2}}$.

TABLE 2. Comparison of Execution Time in Milliseconds

GCD = 1

R	1 - 1000				1 - 500			500 - 1000		
N	10	50	100	250	10	50	100	10	50	100
Blankinship	6	65	239	1357	5	65	233	7	74	255
Algorithm 1	5	9	16	35	2	9	11	13	14	26
Algorithm 2	48	29	165	69	18	59	33	669	2110	3942
Combination	8	27	57	68	9	10	33	14	20	24
Blankinship Without x	2	4	8	13	2	5	6	2	7	13
Algorithm 2 Without x	43	22	141	45	17	49	25	610	2002	3571
Combination Without x	4	6	33	42	5	7	24	2	7	11

TABLE 2. (Continued)

GCD = 2

R	1 - 1000				1 - 500			500 - 1000		
N	10	50	100	250	10	50	100	10	50	100
Blankinship	11	136	477	2844	9	133	474	9	140	492
Algorithm 1	3	9	16	23	3	8	12	8	14	20
Algorithm 2	69	280	250	93	139	62	59	1211	4934	9140
Combination	5	30	92	88	4	41	54	12	25	34
Blankinship Without x	6	73	253	1498	5	70	251	5	74	257
Algorithm 2 Without x	62	251	200	73	125	53	47	1100	4376	8284
Combination Without x	4	21	21	26	3	5	11	6	14	21

TABLE 2. (Continued)

GCD = 3

R	1 - 1000				1 - 500			500 - 1000		
N	10	50	100	250	10	50	100	10	50	100
Blankinship	8	134	471	2853	8	128	479	10	142	485
Algorithm 1	2	8	12	34	3	5	15	5	12	20
Algorithm 2	104	218	67	225	104	21	186	1640	6920	14313
Combination	3	59	62	207	3	20	169	4	44	20
Blankinship Without x	4	72	251	1501	5	69	254	5	74	256
Algorithm 2 Without x	93	192	54	179	92	7	159	1496	6278	13306
Combination Without x	2	52	32	134	2	6	34	2	38	12

5. Applications of the Greatest Common Divisor

The solution of many problems require the associated x_i values (e.g., $GCD = \sum_{i=1}^{n} a_i x_i$) as well as the GCD. Some examples of these applications, such as the Problem of Chinese Remainders, are discussed in [1].

Use of the GCD computation in mathematical programming is demonstrated in solving integer programming problems. Many integer programming problems may be solved using the following algorithm.

Once the linear programming solution is obtained by the simplex method the problem may be written in the form

$$\text{minimize } Z' + \sum_{j\epsilon\overline{B}} c_j x_j$$

$$\text{subject to } x_i + \sum_{j\epsilon\overline{B}} \alpha_j x_j = h_i, \; i\epsilon B \qquad (10)$$

$$x_j \geq 0 \text{ and integer for all } j$$

where $h_i \geq 0$, $c_j \geq 0$, B is the set of basic variables and $\overline{B}$ is the set of non basic variables. If h_i for all i are integer then $x_i = h_i$ for all $i \epsilon B$, is the optimal integer solution. If any of the h_i are fractional then the problem (10) may be further reduced to the form

$$\text{minimize } \sum_{j\epsilon\overline{B}} c'_j x_j$$

$$\text{subject to } \sum_{j\epsilon\overline{B}} a_j x_j \equiv b \pmod{D} \qquad (11)$$

$$x_j \geq 0 \text{ and integer, for all } j$$

where $c'_j = Dc_j$.

The constraint of this problem (11) may be determined by the following procedure:

1. Express all elements of the tableau as a fraction where the numerator is an integer and the denominator is D, the product of the pivot elements. Go to 2.
2. For each row of the tableau compute the GCD of the non zero numerators and D. Go to 3.
3. Select the row, R, with the minimum GCD. If the minimum GCD is unity go to 5. Otherwise go to 4.
4. Compute G, the GCD of the greatest common divisors of the rows. If G is greater than 1 reduce D to D/G. Go to 5.
5. Generate the constraint congruence of (11) by taking row R modulo D. Stop.

Many algorithms [3], [4], and [5] have been developed to solve integer programs once they are in the form of (11).

We use, as an example to illustrate this procedure, problem 3 of the IBM test problems given in [6].

$$\text{Minimize } Z = 13x_1 + 15x_2 + 14x_3 + 11x_4$$

$$\text{Subject to} \quad 4x_1 + 5x_2 + 3x_3 + 6x_4 \geq 96$$

$$20x_1 + 21x_2 + 17x_3 + 12x_4 \geq 200$$

$$11x_1 + 12x_2 + 12x_3 + 7x_4 \geq 101$$

$$x_i \geq 0 \text{ and integer for all } i.$$

The non integer solution from the simplex tableau is:

$$\text{minimize} \quad \frac{12944}{72} + \frac{46}{72} x_2 + \frac{238}{72} x_3 + \frac{64}{72} x_5 + \frac{34}{72} x_6$$

$$\text{subject to} \quad x_7 - \frac{26}{72} x_2 - \frac{194}{72} x_3 - \frac{8}{72} x_5 - \frac{38}{72} x_6 = \frac{1096}{72}$$

$$x_1 + \frac{66}{72} x_2 + \frac{66}{72} x_3 + \frac{12}{72} x_5 - \frac{6}{72} x_6 = \frac{48}{72}$$

$$x_4 + \frac{16}{72} x_2 - \frac{8}{72} x_3 - \frac{30}{72} x_5 + \frac{4}{72} x_6 = \frac{1120}{72}$$

and $D = 72$.

The greatest common divisors of the rows of the tableau are 2, 2, 6, and 4. We may arbitrarily choose between row 1 and row 2. In this problem let $R = 2$.

Compute $G = \text{GCD}(2, 2, 6, 4) = 2$.

The reduced $D = \frac{72}{2} = 36$. The congruence generated by row 2 is:

$23 x_2 + 11 x_3 + 32 x_5 + 17 x_6 \equiv 8 \pmod{36}$.

Therefore the problem reduced to the form of (11) is:

$$\text{minimize } 23 x_2 + 119 x_3 + 32 x_5 + 17 x_6$$

$$\text{subject to } 23 x_2 + 11 x_3 + 32 x_5 + 17 x_6 \equiv 8 \pmod{36}$$

which may be solved by the algorithm outlined in [3] to produce the following solution:

$Z_o = 187$, $X_o = (0, 0, 0, 17, 6, 4, 18)$.

BIBLIOGRAPHY

1. Dickson, L. E., History of the Theory of Numbers. 3 vols. G. E. Stechert and Company, New York, 1934, vol. II, pp 41-99.

2. Blankinship, W. A., "A New Version of the Euclidean Algorithm," American Mathematical Monthly, 70: 742-45, 1963.

3. Greenberg, H., "An Integer Programming Algorithm Using Dynamic Programming," submitted for publication.

4. Gomery, R. E., "On the Relation Between Integer and Non Integer Solutions to Linear Programs," Proceedings National Academy of Science, 53: 260-65, 1965.

5. Shapiro, J. F., "Dynamic Programming Algorithms for the Integer Programming Problem--I: The Integer Programming Problem Viewed as a Knapsack Type Problem," Operations Research, 16: 103-21, 1968.

6. Haldi, J., "25 Integer Programming Test Problems," Working Paper No. 43, Stanford University, Palo Alto, California, Dec. 1964, pp. 12-13.

7. Griffin, H., Elementary Theory of Numbers, McGraw-Hill Book Company, Inc., New York, 1954, pp 9-10.

Appendix I

FORTRAN IV Program of the Blankinship Method

```
      DIMENSION NR(251), NRS(251), NX(251,251)
C 'N' IS THE NUMBER OF INTEGERS
C 'NR(I)' IS THE ARRAY OF INTEGERS - WILL BE
C ALTERED BY PROGRAM
      DO 600 IJK = 1, 30
      READ (5, 1 ) N, MULT, IAD, IFUD, IX,IR1, IR2
    1 FORMAT(4I5,3I10)
      KNT = 0
  750 KNT = KNT + 1
      KMA = IX
C GENERATION OF RANDOM NUMBERS USING RANDU ROUTINE
C 'NRS(I)' IS THE ARRAY OF INTEGERS - WILL NOT BE ALTERED
      DO 59 I = 1,N
      CALL RANDU (IX, IY, YFL)
      IX = IY
      MMM = YFL * MULT
      NR(I) = (MMM + IAD) * IFUD
      NRS( I) = NR(I)
   59 CONTINUE
C EXECUTION TIMED FROM THIS POINT
C ESTABLISHING AN IDENTITY MATRIX
      DO 200 I = 1,N
      DO 200 J = 1,N
      IF(I.EQ.J) GO TO 201
C 'NX(I,J)' IS TH MATRIX OF 'X' VALUES
      NX(I,J) = 0
      GO TO 200
  201 NX(I,J) = 1
  200 CONTINUE
C 'KONT' IS AN ITERATION COUNTER
C 'J' IS THE INDEX OF THE OPERATOR
C 'MIN' IS THE MINUMUM OF THE ROW LEADERS
      KONT = 0
      J = 1
      MIN = NR(1)
C DETERMINING THE MINIMUM ROW LEADER (OPERATOR)
   15 DO 20 I = 1, N
      IF (MIN .LE. NR(I)) GO TO 20
      MIN = NR(I)
      J = I
   20 CONTINUE
      IF(MIN.EQ. 1) GO TO 50
      KONT = KONT + 1
C DETERMINING A NONZERO ROW LEADER (OPERAND)
      DO 30 I = 1, N
      IF ( NR(I) .EQ. 5000) GO TO 30
C NOTE BELOW USE OF 5000 TO DENOTE ZERO
```

```
      IF (J .EQ. I) GO TO 30
      JJ = I
C 'JJ' IS THE INDEX OF THE OPERAND

      GO TO 40
   30 CONTINUE
C NOTE NATURAL EXIT OF THIS DO LOOP INDICATES
C COMPLETION OF PROCEDURE

      GO TO 50
C COMPUTATION OF REMAINDER AND NEW ROW LEADER

   40 Z = NR(JJ) / MIN
      IQ = Z
C 'IQ' IS THE GREATEST INTEGER

      NR(JJ) = NR(JJ) - IQ * MIN
      IF( NR(JJ) .NE. 0) GO TO 202
C FOR PROGRAMMING   LOGIC  A ROW LEADER WITH ZERO VALUE
C IS SET AT 5000

      NR(JJ) = 5000
C PERFORM NEXT ITERATION - ROW LEADER IS ZERO
C 'X' VALUES NOT REQUIRED

      GO TO 15
C COMPUTATION OF NEW ROW

  202 DO 203 I = 1, N
      NX(JJ,I) = NX(JJ,I) - NX(J,I) * IQ
  203 CONTINUE
C PERFORM NEXT ITERATION

      GO TO 15
   50 CONTINUE
C END OF EXECUTION TIMING

      WRITE(6, 100)
  100 FORMAT(1H1,45X,'THE GREATEST COMMON DIVISOR OF ;'
     1//)
      WRITE (6, 101) (NRS(I), I = 1, N)
  101 FORMAT (20X, 10I8, //)
      WRITE(6,102) MIN
  102 FORMAT(/////, 55X, 'IS ', I4)
      WRITE (6,103) KONT
  103 FORMAT(///,50X,I5,'  ITERATIONS USED')
      WRITE(6,205) (NX(J ,I) ,I = 1,N)
  205 FORMAT(///,10X, 10I10)
      WRITE (6,502) N, IFUD, IR1, IR2
  502 FORMAT(//,32X,I5,' NUMBERS, MULTIPLES OF',I3,
     1' OVER THE RANGE',I4, ' -',I5)
      IX = KMA/ 3
      IF(KNT.NE.3) GO TO 750
  600 CONTINUE
      END
```

Appendix II

FORTRAN IV Program of the Blankinship Method without X_i

```
      DIMENSION  NR(1000), NRS(1000)
C 'N' IS THE NUMBER OF INTEGERS
C 'NR(I)' IS THE ARRAY OF INTEGERS - WILL BE
C ALTERED BY THE PROGRAM
C 'NRS(I)' IS THE ARRAY OF INTEGERS - WILL NOT BE ALTERED
      DO 600 IJK = 1, 30
      READ (5, 1 ) N, MULT, IAD, IFUD, IX,IR1, IR2
    1 FORMAT(4I5,3I10)
      KNT = 0
  750 KNT = KNT + 1
      KMA = IX
C GENERATION OF RANDOM NUMBERS USING RANDU ROUTINE
      DO 59 I = 1,N
      CALL RANDU (IX, IY, YFL)
      IX = IY
      MMM = YFL * MULT
      NR(I) = (MMM + IAD) * IFUD
      NRS(I) = NR(I)
   59 CONTINUE
C EXECUTION TIMED FROM THIS POINT
      KONT = 0
      J = 1
C DETERMINING THE MINIMUM ROW LEADER (OPERATOR)
C 'MIN' IS THE MINUMUM OF THE ROW LEADERS
      MIN = NR(1)
   15 DO 20 I = 1, N
      IF (MIN .LE. NR(I)) GO TO 20
      MIN = NR(I)
      J = I
C 'J' IS THE INDEX OF THE OPERATOR
   20 CONTINUE
      IF (MIN .EQ. 1) GO TO 50
C 'KONT' IS AN ITERATION COUNTER
      KONT = KONT + 1
C DETERMINING A NONZERO ROW LEADER (OPERAND)
      DO 30 I = 1, N
      IF ( NR(I) .EQ. 5000) GO TO 30
C NOTE BELOW USE OF 5000 TO DENOTE ZERO
      IF (J .EQ. I) GO TO 30
C 'JJ' IS THE INDEX OF THE OPERAND
      JJ = I
      GO TO 40
C NOTE NATURAL EXIT OF THIS DO LOOP INDICATES
C COMPLETION OF PROCEDURE
```

```
   30 CONTINUE
      GO TO 50
C COMPUTATION OF REMAINDER AND NEW ROW LEADER
   40 Z = NR(JJ) / MIN
      IQ = Z
C 'IQ' IS THE GREATEST INTEGER
      NR(JJ) = NR(JJ) - IQ * MIN
      IF( NR(JJ) .NE. 0) GO TO 15
C FOR PROGRAMMING   LOGIC  A ROW LEADER WITH ZERO VALUE
C IS SET AT 5000
      NR(JJ) = 5000
C PERFORM NEXT ITERATION
      GO TO 15
C END OF EXECUTION TIMING
   50 CONTINUE
      WRITE(6, 100)
  100 FORMAT(1H1,45X,'THE GREATEST COMMON DIVISOR OF :'
     1//)
      WRITE (6, 101) (NRS(I), I = 1, N)
  101 FORMAT (20X, 10I8, //)
      WRITE(6,102) MIN
  102 FORMAT(/////, 55X, 'IS ', I4)
      WRITE (6,103) KONT
  103 FORMAT(///,50X,I5,'  ITERATIONS USED')
      WRITE (6,502) N, IFUD, IR1, IR2
  502 FORMAT(//,32X,I5,' NUMBERS, MULTIPLES OF',I3,
     1' OVER THE RANGE',I4, ' -',I5)
      IX = KMA / 3
      IF(KNT.NE.3) GO TO 750
  600 CONTINUE
      END
```

Appendix III

FORTRAN IV Program of the Algorithm 1 Method

```
      DIMENSION  IR(100), NR(100,255), ID(100),IX(255)
C 'N' IS THE NUMBER OF INTEGERS
C NR(1,I) IS THE ARRAY OF INTEGERS
C GENERATION OF RANDOM NUMBERS USING RANDU ROUTINE

      DO 600 IJK = 1, 30
      READ(5,1) N, MULT, IAD, IFUD, LX, IR1, IR2
    1 FORMAT(4I5,3I10)
      KNT = 0
  750 KNT = KNT + 1
      KMA = LX

C GENERATION OF RANDOM NUMBERS USING RANDU ROUTINE

      DO 59 I = 1,N
      CALL RANDU (LX, IY, YFL)
      LX = IY
      MMM = YFL * MULT
      NR(1,I) = (MMM + IAD) * IFUD
   59 CONTINUE
      DO 2 I = 1, N
      IX(I) = 0
    2 CONTINUE

C EXECUTION TIMED FROM THIS POINT

      M = 1
      IR(1) =1

C DETERMINATION OF MINIMUM VALUED INTEGER
C 'MIN' IS THE MINIMUM VALUED INTEGER

      MIN = NR(1,1)
      DO 10 I=2,N
      IF(MIN.LE.NR(1,I)) GO TO 10
      MIN = NR(1,I)

C ' IR '  IS THE INDEX OF THE MINIMUM VALUED INTEGER

      IR(1) = I
   10 CONTINUE
      ID(1) = MIN
      IF(MIN.EQ.1) GO TO 12
      DO 3 I = 1, N
      IG = NR(1,I) / MIN
      IREM = NR(1,I) - IG * MIN
      IF (IREM .EQ. 0) GO TO 3
      GO TO 11
    3 CONTINUE
      GO TO 12
   11 NMIN = ID(M)
      ISTOP = 0

C THIS LOOP CALCULATES THE NR(M,I) MODULO D AND SELECTS
C THE MINIMUM OF THE NEW ROW  (NMIN)

      DO 15 I =1,N
      NUM =NR(M,I)
      IF(NUM.EQ.0) GO TO 14
      IG= NUM / ID(M)
      IREM =NUM - IG* ID(M)
      IF(IREM.EQ.0) GO TO 14
```

```
      NR(M+1,I) = IREM
      ISTOP= 1
      IF (NMIN .LE. IREM) GO TO 15
      NMIN = IREM
      IR(M +1) = I
      GO TO 15
   14 NR(M+1,I) = 0
   15 CONTINUE
      IDM =ID(M)
C DETERMINATION OF THE GCD OF THE NEW MIN AND D
   17 IG = IDM/NMIN
      IREM = IDM - IG * NMIN
      IF(IREM.EQ.0) GO TO 16
      IDM = NMIN
      NMIN = IREM
      GO TO 17
   16 M = M +1
      ID(M)=NMIN
C 'ISTOP' EQUAL TO ZERO INDICATES GCD IS DETERMINED
      IF(ISTOP.NE.0) GO TO 11
      M = M - 1
C 'IGCD' IS THE GREATEST COMMON DIVISOR
      IGCD = ID(M)
C DETERMINATION OF X VALUES
      MM = M
      J = IR(M)
      CALL GETX(IGCD, NR(M,J),ID(M-1),IX(J))
      MM2 = MM-2
      IF(M.EQ.2) GO TO 20
      DO 24 I =1,MM2
      ISUM = 0
      M = MM-I
      J = IR(M)
      M1 = M+1
      DO 25 II = M1, MM
      K = IR(II)
      ISUM = ISUM +NR(M,K)* IX(K)
   25 CONTINUE
      IZ = IGCD - ISUM
      IF(IZ) 26, 27, 28
   26 IZ = IZ + ID(M-1)
      GO TO 28
   27 IX(J) = 0
      GO TO 24
   28 CALL GETX (IZ, NR(M,J),ID(M-1), IX(J))
   24 CONTINUE
   20 M = M- 1
      J = IR(M)
      ISUM = 0
      DO 30 I = 2,MM
      K = IR(I)
      ISUM = ISUM + NR(M,K) * IX(K)
   30 CONTINUE
      IX(J) =(IGCD - ISUM)/ NR(M,J)
      GO TO 31
   12 I = IR(1)
      IX(I) = 1
C END OF EXECUTION TIMING
   31 CONTINUE
      WRITE(6,100)
```

```
100 FORMAT(1H1,45X,'THE GREATEST COMMON DIVISOR OF :'
   1//)
    WRITE (6,101) (NR(1,I),I = 1,N)
101 FORMAT(20X,10I8,//)
    WRITE (6,102) IGCD
102 FORMAT(/////,55X,'IS ', I4)
    WRITE(6,110) (IX(I), I= 1,N)
110 FORMAT(//,10X, 10I10)
    WRITE (6,502) N, IFUD, IR1, IR2
502 FORMAT(//,32X,I5,' NUMBERS, MULTIPLES OF',I3,
   1' OVER THE RANGE',I4, ' -',I5)
    LX = KMA /  3
    IF(KNT.NE.3) GO TO 750
600 CONTINUE
    END


    SUBROUTINE GETX(MIN,IB,NR1,IXJ)
C SOLVES CONGRUENCES OF THE FORM IB*IX = MIN(MOD NR1)
    IF(MIN.EQ.IB) GO TO 2
    IB1 = IB
    JFM = 1
    DO 5 I = 1, NR1
    JFM = JFM +1
    IB1 = IB1 + IB
    IF(NR1 - IB1) 6,7,7
  6 IB1 = IB1 - NR1
  7 IF(MIN.EQ.IB1) GO TO 8
  5 CONTINUE
  8 IXJ = JFM
    GO TO 9
  2 IXJ = 1
  9 RETURN
    END
```

Appendix IV

FORTRAN IV Program of the Algorithm 2 Method

```
      DIMENSION NR(1000), IB(2000), JF(1000), ID(1000)
     1IDX(2000),IS(2000),IX(1001)
C 'N' IS THE NUMBER OF INTEGERS
C 'NR(I)' IS THE ARRAY OF INTEGERS
      DO 600 IJK = 1, 30
      READ (5, 1 ) N, MULT, IAD, IFUD, LX,IR1, IR2
    1 FORMAT(4I5,3I10)
      KNT = 0
  750 KNT = KNT + 1
      KMA = LX
C GENERATION OF RANDOM NUMBERS USING RANDU ROUTINE
      DO 59 I = 1,N
      CALL RANDU (LX, IY, YFL)
      LX = IY
      MMM = YFL * MULT
      NR(I) = (MMM + IAD) * IFUD
   59 CONTINUE
C EXECUTION TIMED FROM THIS POINT
C INTIALIZING THE VALUES OF PROGRAM VARIABLES
      KONT = 0
      LL = 0
      II = 0
      N1 = N + 1
      DO 35 I = 1, N1
C 'IX(I)' ARE THE ACTUAL X VALUES - THEY ARE
C INITIALIZED TO ZERO HERE
      IX(I) = 0
   35 CONTINUE
C 'MIN' IS THE MINUMUM VALUED INTEGER
C 'IHOLD' IS THE INDEX OF THE MINIMUM VALUED INTEGER
      MIN = NR(1)
      IHOLD = 1
C DETERMINING THE MINIMUM
      DO 10 I = 2,N
      IF(MIN.LE.NR(I)) GO TO 10
      MIN = NR(I)
      IHOLD = I
   10 CONTINUE
      L = 0
C DETERMINATION OF 'B' ARRAY
C DETERMINATION  OF MOD VALUES
      DO 11 I = 1, N
      A = NR(I)
      B = MIN
      C = A/B
C 'IG' IS THE GREATEST INTEGER
      IG = C
```

```
      IF(C-IG) 11, 16, 12
C 'L' IS THE COUNTER OF INTEGERS NOT EVENLY
C DIVISIBLE BY THE MINIMUM
   12 L = L +1
      MINIG = MIN * IG
C 'IB(I)' IS AN ARRAY OF B VALUES
      IB(L) = NR(I) - MINIG
      L = L + 1
      IB(L) = MIN + MINIG - NR(I)
      GO TO 11
C 'LL' IS THE COUNTER OF INTEGERS EVENLY
C DIVISIBLE BY THE MINIMUM
C 'IS(LL)' IS AN ARRAY OF INDICES OF INTEGERS WHICH
C ARE EVENLY DIVISIBLE BY THE MINIMUM
   16 LL = LL + 1
      IS(LL) = I
   11 CONTINUE
      K = L
C 'M' IS THE SIZE OF THE 'B' ARRAY
      M = K + 1
      IF(L.NE.0) GO TO 17
      IGCD = MIN
      IX(IHOLD) =1
      IX(N1) = IGCD - 1
      GO TO 50
   17 IB(M) = MIN - 1
C 'JF(I) IS THE ARRAY OF F VALUES
   18 JF(1) = 0
C NOTE PROGRAM LOGIC REQUIRES 'F'ARRAY
C INDICES TO BE INCREASED BY +1
C INITAIALIZING 'F' ARRAY TO MIN PLUS 1
      DO 13 I = 2, MIN
      JF(I) = MIN + 1
   13 CONTINUE
C 'ISTOP' RECORDS NUMBER OF CHANGES OF F VALUES
C IN THE PRESENT ITERATION
C 'KONT' IS AN ITERATION COUNTER
   29 ISTOP = 0
      KONT = KONT + 1
C DETERMINATION OF NEW F VAUES
      DO 14 I = 2, MIN
      IF(JF(I).EQ. 0) GO TO 14
      DO 15 J = 1, K
      IF (I - 1 - IB(J)) 21, 22, 23
C 'JJ' IS THE INDEX OF F
   21 JJ = I - IB(J) + MIN
      GO TO 24
   22 JJ = 1
      GO TO 24
   23 JJ = I - IB(J)
   24 IF ( JF(I) .LE. JF(JJ)) GO TO 15
```

```
C DETERMINATION OF MINIMUM VALUES WHICH ARE NEW F VALUES
      JF (I) = JF(JJ)
C 'ID(I)' IS AN ARRAY OF INDICES OF B VALUES
C ASSOCIATED WITH THE MINIMUM
      ID(I) = J
      ISTOP = ISTOP + 1
   15 CONTINUE
      IF(I-1-IB(M)) 25, 26, 27
   25 JJ = I- IB(M) + MIN
      GO TO 28
   26 JJ = 1
      GO TO 28
   27 JJ = I - IB(M)
   28 IF(JF(I) .LE. JF(JJ) + 1) GO TO 14
      JF(I) = JF(JJ) + 1
      ID(I) = M
      ISTOP = ISTOP + 1
   14 CONTINUE
      IF(ISTOP .GT. 0) GO TO 29
C NO CHANGED F VALUES INDICATES PROCEDURE COMPLETE
C 'IGCD' IS THE GREATEST COMMON DIVISOR
C 'IDX(I)' ARE THE X' VALUES ASSOCIATED WITH THE B VALUES
      IGCD = JF(2) + 1
      DO 36 I = 1, M
      IDX(I) = 0
   36 CONTINUE
      NN = 1
   42 MM = ID(NN+1)
C CALCULATION OF X' VALUES
      IDX(MM) = IDX(MM) + 1
      NN = NN - IB(MM)
      IF(NN) 40, 41, 42
   40 NN = NN+ MIN
      GO TO 42
   41 II = 1
      IDUM = 1
C 'IDUM','LDUM', AND 'II' ARE DUMMY VARIABLES
C USED FOR INCREMENTING THE INDICES
C DETERMINATION OF X VALUES FROM X' VALUES
      DO 43 J =  1,LL
      IF(IS(J) .EQ.IDUM) GO TO 44
      LDUM = IS(J) -1
      DO 45 I = IDUM, LDUM
      IX(I) = IDX(II) - IDX(II+1)
      II = II + 2
   45 CONTINUE
      IDUM = LDUM + 1
   44 IX(IDUM) = 0
      IDUM = IDUM + 1
   43 CONTINUE
      IF(IDUM - 1.EQ.N) GO TO 48
      DO 46 I = IDUM, N
      IX(I) = IDX(II) - IDX(II + 1)
      II = II + 2
   46 CONTINUE
   48 IX(N+1) = IDX(II)
      ISUM = 0
C ALGEBRAIC COMPUTATION OF REMAINING X VALUE
      DO 47 I = 1, N
```

```
      ISUM = ISUM + NR(I) * IX(I)
   47 CONTINUE
      IX(IHOLD) = (1 + IX(N+1) - ISUM) / MIN

C END OF EXECUTION TIMING

   50 CONTINUE
      WRITE (6, 100)
  100 FORMAT(1H1,45X,'THE GREATEST COMMON DIVISOR OF ;'
     1//)
      WRITE (6,101) (NR(I) , I = 1,N)
  101 FORMAT(20X,10I8,//)
      WRITE (6,102) IGCD
  102 FORMAT(/////,55X,'IS ', I4)
      WRITE(6,103) MIN,M, KONT
  103 FORMAT(//,32X,'THE F MATRIX IS ',I5,' BY ',I5,'.',
     15X,I4,'  ITERATIONS USED')
      WRITE(6, 110) (IX(I), I = 1, N1)
  110 FORMAT ( //, 10X, 10I10)
      WRITE (6, 111) II, M
  111 FORMAT (/////, 10X, 2I10)
      WRITE (6,502) N, IFUD, IR1, IR2
  502 FORMAT(//,32X,I5,' NUMBERS, MULTIPLES OF',I3,
     1' OVER THE RANGE',I4, ' -',I5)
      LX = KMA /3
      IF(KNT.NE.3) GO TO 750
  600 CONTINUE
      END
```

Appendix V

FORTRAN IV Program of the Algorithm 2 Method without X_i

```
      DIMENSION NR(1000),  IB(1000), JF(1000)
C 'N' IS THE NUMBER OF INTEGERS
C 'NR(I)' IS THE ARRAY OF INTEGERS
      DO 600 IJK = 1, 30
      READ (5, 1 ) N, MULT, IAD, IFUD, IX,IR1, IR2
    1 FORMAT(4I5,3I10)
      KNT = 0
  750 KNT = KNT + 1
      KMA = IX
C GENERATION OF RANDOM NUMBERS USING RANDU ROUTINE
      DO 59 I = 1,N
      CALL RANDU (IX, IY, YFL)
      IX = IY
      MMM = YFL * MULT
      NR(I) = (MMM + IAD) * IFUD
   59 CONTINUE
C EXECUTION TIMED FROM THIS POINT
C 'MIN' IS THE MINUMUM VALUED INTEGER
C DETERMINING THE MINIMUM
      KONT = 0
      MIN = NR(1)
      DO 10 I = 2,N
      IF(MIN.LE.NR(I)) GO TO 10
      MIN = NR(I)
   10 CONTINUE
      L = 0
C DETERMINATION OF MOD VALUES AND 'B' ARRAY
      DO 11 I = 1, N
      A = NR(I)
      B = MIN
      C = A/B
C 'IG' IS THE GREATEST INTEGER
C 'L' IS THE COUNTER OF INTEGERS NOT EVENLY
C DIVISIBLE BY THE MINIMUM
      IG = C
      IF(C-IG) 11, 11, 12
   12 L = L +1
      MINIG = MIN * IG
C 'IB(I)' IS AN ARRAY OF B VALUES
      IB(L) = NR(I) - MINIG
      L = L + 1
      IB(L) = MIN + MINIG - NR(I)
   11 CONTINUE
      K = L
      M = K + 1
      IF(L.NE.0) GO TO 16
      IGCD = MIN
      GO TO 50
   16 IB(M) = MIN - 1
C 'JF(I) IS THE ARRAY OF F VALUES
```

```
C NOTE PROGRAM LOGIC REQUIRES 'F'ARRAY
C INDICES INCREASED BY +1
C INITAIALIZING 'F' ARRAY TO MIN PLUS 1

      JF(1) = 0
      DO 13 I = 2, MIN
      JF(I) = MIN + 1
   13 CONTINUE

C 'ISTOP' RECORDS NUMBER OF CHANGES OF F VALUES
C IN THE PRESENT ARRAY
C 'KONT' IS AN ITERATION COUNTER
C DETERMINATION OF NEW F VAUES

   29 ISTOP = 0
      KONT = KONT + 1
      DO 14 I = 2, MIN
      IF(JF(I).EQ. 0) GO TO 14
      DO 15 J = 1, K
      IF (I - 1 - IB(J)) 21, 22, 23
   21 JJ = I - IB(J) + MIN
      GO TO 24
   22 JJ = 1
      GO TO 24
   23 JJ = I - IB(J)

C DETERMINATION OF MINIMUM VALUES WHICH ARE NEW F VALUES

   24 IF ( JF(I) .LE. JF(JJ)) GO TO 15
      JF (I) = JF(JJ)
      ISTOP = ISTOP + 1
   15 CONTINUE
      IF(I-1-IB(M)) 25, 26, 27
   25 JJ = I- IB(M) + MIN
      GO TO 28
   26 JJ = 1
      GO TO 28
   27 JJ = I - IB(M)
   28 IF(JF(I) .LE. JF(JJ) + 1) GO TO 14
      JF(I) = JF(JJ) + 1
      ISTOP = ISTOP + 1
   14 CONTINUE
      IF(ISTOP .GT. 0) GO TO 29

C NO CHANGED F VALUES INDICATES PROCEDURE COMPLETE
C 'IGCD' IS THE GREATEST COMMON DIVISOR

      IGCD = JF(2) + 1

C END OF EXECUTION TIMING

   50 CONTINUE
      WRITE (6, 100)
  100 FORMAT(1H1,45X,'THE GREATEST COMMON DIVISOR OF :'
     1//)
      WRITE(6,103) MIN,M, KONT
  103 FORMAT(//,32X,'THE F MATRIX IS ',I5,' BY ',I5,'.',
     15X,I4,'  ITERATIONS USED')
      WRITE (6,502) N, IFUD, IR1, IR2
  502 FORMAT(//,32X,I5,' NUMBERS, MULTIPLES OF',I3,
     1' OVER THE RANGE',I4, ' -',I5)
      IX = KMA / 3
      IF(KNT.NE.3) GO TO 750
  600 CONTINUE
      END
```

Appendix VI

FORTRAN IV Program of the Combination Method

```
      DIMENSION JF(1000), ID(1000), IDX(2000)
      COMMON NR(1000), IB(2000), IS(2000), L, LL, N, IJ,
     1IHOLD, IX(1001)

C 'N' IS THE NUMBER OF INTEGERS
C 'NR(I)' IS THE ARRAY OF INTEGERS

      DO 600 IJK = 1, 30
      READ (5, 1 ) N, MULT, IAD, IFUD, LX,IR1, IR2
    1 FORMAT(4I5,3I10)
      KNT = 0
  750 KNT = KNT + 1
      KMA = LX

C GENERATION OF RANDOM NUMBERS USING RANDU ROUTINE

      DO 59 I = 1,N
      CALL RANDU (LX, IY, YFL)
      LX = IY
      MMM = YFL * MULT
      NR(I) = (MMM + IAD) * IFUD
   59 CONTINUE

C EXECUTION TIMED FROM THIS POINT

      KONT = 0
      N2 = 0
      II = 0
      N1 = N + 1
      DO 35 I = 1, N1

C 'IX(I)' ARE THE ACTUAL X VALUES - THEY
C ARE INITIALIZED TO ZERO HERE

      IX(I) = 0
   35 CONTINUE

C 'MIN' IS THE MINUMUM VALUED INTEGER
C 'IHOLD' IS THE INDEX OF THE MINIMUM VALUED INTEGER
C DETERMINING THE MINIMUM

      MIN = NR(1)
      IHOLD = 1
      DO 10 I = 2,N
      IF(MIN.LE.NR(I)) GO TO 10
      MIN = NR(I)
      IHOLD = I
   10 CONTINUE
      AN = N
      KUTOFF = (SQRT(AN)) * 1.5
      IF(MIN.LE.KUTOFF) GO TO 699
      N2 = 1
      DO 700 I=1,N
      IF(NR(I) .EQ.MIN) GO TO 700
      IJ = I
      GO TO 701
  700 CONTINUE
  701 NUM = NR(IJ)
  710 IG = NUM/ MIN
      IREM = NUM - IG * MIN
      IF(IREM.EQ.0) GO TO 699
      NUM = MIN
      MIN = IREM
```

```
      GO TO 710
  699 CONTINUE
      CALL BARRAY(MIN)
      K = L
C 'M' IS THE SIZE OF THE 'B' ARRAY
      M = K + 1
      IF(L.NE.0) GO TO 17
      IGCD = MIN
      IF(N2.NE.0) GO TO 19
      IX(IHOLD) =1
      IX(N1) = IGCD - 1
      GO TO 50
   19 IX(N1) = IGCD - 1
      A = NR(IJ)
      B = NR(IHOLD)
      C = A/B
      IG = C
      IB(1) = NR(IJ) - IG * NR(IHOLD)
      IF(MIN.LT.IB(1)) GO TO 51
      IX(IJ) = MIN/ IB(1)
      GO TO 54
   51 CALL GETX (MIN)
      GO TO 54
   17 IB(M) = MIN - 1
   18 JF(1) = 0
C 'JF(I) IS THE ARRAY OF F VALUES
C NOTE PROGRAM LOGIC REQUIRES 'F'ARRAY
C INDICES INCREASED BY +1
C INITAIALIZING 'F' ARRAY TO MIN PLUS 1
      DO 13 I = 2, MIN
      JF(I) = MIN + 1
   13 CONTINUE
C 'ISTOP' RECORDS NUMBER OF CHANGES OF F VALUES
C IN THE PRESENT ITERATION
C 'KONT' IS AN ITERATION COUNTER
   29 ISTOP = 0
      KONT = KONT + 1
C DETERMINATION OF NEW F VAUES
      DO 14 I = 2, MIN
      IF(JF(I).EQ. 0) GO TO 14
      DO 15 J = 1, K
      IF (I - 1 - IB(J)) 21, 22, 23
C 'JJ' IS THE INDEX OF F
   21 JJ = I - IB(J) + MIN
      GO TO 24
   22 JJ = 1
      GO TO 24
   23 JJ = I - IB(J)
   24 IF ( JF(I) .LE. JF(JJ)) GO TO 15
C DETERMINATION OF MINIMUM VALUES WHICH ARE NEW F VALUES
      JF (I) = JF(JJ)
C 'ID(I)' IS AN ARRAY OF INDICES OF B VALUES
C ASSOCIATED WITH THE MINIMUM
      ID(I) = J
      ISTOP = ISTOP + 1
   15 CONTINUE
```

```
      IF(I-1-IB(M)) 25, 26, 27
   25 JJ = I- IB(M) + MIN
      GO TO 28
   26 JJ = 1
      GO TO 28
   27 JJ = I - IB(M)
   28 IF(JF(I) .LE. JF(JJ) + 1) GO TO 14
      JF(I) = JF(JJ) + 1
      ID(I) = M
      ISTOP = ISTOP + 1
   14 CONTINUE
      IF(ISTOP .GT. 0) GO TO 29

C NO CHANGED F VALUES INDICATES PROCEDURE COMPLETE
C 'IGCD' IS THE GREATEST COMMON DIVISOR
C 'IDX(I)' ARE THE X' VALUES ASSOCIATED WITH THE B VALUES

      IGCD = JF(2) + 1
      DO 36 I = 1, M
      IDX(I) = 0
   36 CONTINUE
      NN = 1
   42 MM = ID(NN+1)

C CALCULATION OF X' VALUES

      IDX(MM) = IDX(MM) + 1
      NN = NN - IB(MM)
      IF(NN) 40, 41, 42
   40 NN = NN+ MIN
      GO TO 42
   41 II = 1
      IDUM = 1

C 'IDUM','LDUM', AND 'II' ARE DUMMY VARIABLES
C USED FOR INCREMENTING THE INDICES
C DETERMINATION OF X VALUES FROM X' VALUES

      DO 43 J =  1,LL
      IF(IS(J) .EQ.IDUM) GO TO 44
      LDUM = IS(J) -1
      DO 45 I = IDUM, LDUM
      IX(I) = IDX(II) - IDX(II+1)
      II = II + 2
   45 CONTINUE
      IDUM = LDUM + 1
   44 IX(IDUM) = 0
      IDUM = IDUM + 1
   43 CONTINUE
      IF(IDUM - 1.EQ.N) GO TO 48
      DO 46 I = IDUM, N
      IX(I) = IDX(II) - IDX(II + 1)
      II = II + 2
   46 CONTINUE
   48 IX(N+1) = IDX(II)
      IF(N2.EQ.0) GO TO 54
      CALL GETBS (IGCD)
   54 ISUM = 0

C ALGEBRAIC COMPUTATION OF REMAINING X VALUE

      DO 47 I = 1, N
      ISUM = ISUM + NR(I) * IX(I)
   47 CONTINUE
      IX(IHOLD) = (1+ IX(N+1) - ISUM) / NR(IHOLD)

C END OF EXECUTION TIMING

   50 CONTINUE
      WRITE (6, 100)
```

```
100 FORMAT(1H1,45X,'THE GREATEST COMMON DIVISOR OF :'
   1//)
    WRITE (6,101) (NR(I) , I = 1,N)
101 FORMAT(20X,10I8,//)
    WRITE (6,102) IGCD
102 FORMAT(/////,55X,'IS ', I4)
    WRITE(6,103) MIN,M, KONT
103 FORMAT(//,32X,'THE F MATRIX IS ',I5,' BY ',I5,'.',
   15X,I4,'  ITERATIONS USED')
    WRITE(6, 110) (IX(I), I = 1, N1)
110 FORMAT ( //, 10X, 10I10)
    WRITE (6, 111) II, M
111 FORMAT (/////, 10X, 2I10)
    WRITE (6,502) N, IFUD, IR1, IR2
502 FORMAT(//,32X,I5,' NUMBERS, MULTIPLES OF',I3,
   1' OVER THE RANGE',I4, ' -',I5)
    LX = KMA /3
    IF(KNT.NE.3) GO TO 750
600 CONTINUE
    END


    SUBROUTINE BARRAY(MIN)
    COMMON NR(1000) , IB(2000) , IS(2000) , L , LL , N
    L = 0
    LL = 0

C DETERMINATION OF 'B' ARRAY
C DETERMINIATION OF MOD VALUES

    DO 11 I = 1, N
    A = NR(I)
    B = MIN
    C = A/B

C 'IG' IS THE GREATEST INTEGER

    IG = C
    IF(C-IG) 11, 16, 12

C 'L' IS THE COUNTER OF INTEGERS NOT EVENLY
C DIVISIBLE BY THE MINIMUM

 12 L = L +1
    MINIG = MIN * IG
C 'IB(I)' IS AN ARRAY OF B VALUES
    IB(L) = NR(I) - MINIG
    L = L + 1
    IB(L) = MIN + MINIG - NR(I)
    GO TO 11
 16 LL = LL + 1

C 'LL' IS THE COUNTER OF INTEGERS EVENLY
C DIVISIBLE BY THE MINIMUM
C 'IS(LL)' IS AN ARRAY OF INDICES OF INTEGERS
C EVENLY DIVISIBLE BY THE MINIMUM

    IS(LL) = I
 11 CONTINUE
    RETURN
    END


    SUBROUTINE GETBS (IGCD)

C DETERMINES COEFFICIENTS OF THE REDUCED EQUATION

    COMMON NR(1000), IB(2000), IS(2000), L, LL, N, IJ,
```

```
     1IHOLD, IX(1001)
      DO 11 I = 1,N
      A = NR(I)
      B = NR(IHOLD)
      C = A/B
      IG = C
      IF(C-IG) 11, 16, 12
   12 MINIG = NR(IHOLD) * IG
      IB(I) = NR(I) - MINIG
      GO TO 11
   16 IB(I) = 0
   11 CONTINUE
      ISUM = 0
      DO 47 I = 1,N
      ISUM = ISUM + IB(I) * IX(I)
   47 CONTINUE
      ISUM = IGCD - ISUM
      IF(ISUM) 1,2,3
    1 ISUM = ISUM + NR(IHOLD)
      GO TO 3
    2 IX(IJ) = 0
      GO TO 5
    3 IB(1) = IB(IJ)
      CALL GETX (ISUM)
    5 RETURN
      END


      SUBROUTINE GETX(MIN)
C SOLVES CONGRUENCES OF THE FORM IB*IX = MIN(MOD NR1)
      COMMON NR(1000), IB(2000), IS(2000), L, LL, N, IJ,
     1IHOLD, IX(1001)
      NR1 = NR(IHOLD)
      IF (MIN .EQ. IB(1)) GO TO 2
      IB1 = IB(1)
      JFM = 1
      DO 5 I = 1, NR1
      JFM = JFM +1
      IB1 = IB1 + IB(1)
      IF(NR1 - IB1) 6,7,7
    6 IB1 = IB1 - NR1
    7 IF (MIN .EQ. IB1) GO TO 8
    5 CONTINUE
    8 IX(IJ) = JFM
      GO TO 9
    2 IX(IJ) = 1
    9 RETURN
      END
```

Appendix VII

FORTRAN IV Program of the Combination Method without X_i

```
      DIMENSION NR(1000),  IB(1000), JF(1000)
C 'N' IS THE NUMBER OF INTEGERS
C 'NR(I)' IS THE ARRAY OF INTEGERS
      DO 600 IJK = 1, 30
      READ (5, 1 ) N, MULT, IAD, IFUD, IX,IR1, IR2
    1 FORMAT(4I5,3I10)
      KNT = 0
  750 KNT = KNT + 1
      KMA = IX
C GENERATION OF RANDOM NUMBERS USING RANDU ROUTINE
      DO 59 I = 1,N
      CALL RANDU (IX, IY, YFL)
      IX = IY
      MMM = YFL * MULT
      NR(I) = (MMM + IAD) * IFUD
   59 CONTINUE
C EXECUTION TIMED FROM THIS POINT
C 'MIN' IS THE MINUMUM VALUED INTEGER
C DETERMINING THE MINIMUM
      KONT = 0
      MIN = NR(1)
      DO 10 I = 2,N
      IF(MIN.LE.NR(I)) GO TO 10
      MIN = NR(I)
   10 CONTINUE
      KUTOFF = (SQRT N ) * 1.5
      IF(MIN.LE.KUTOFF) GO TO 699
      DO 700 I=1,N
      IF(NR(I) .EQ.MIN) GO TO 700
      IJ = I
      GO TO 701
  700 CONTINUE
  701 NUM = NR(IJ)
  710 IG = NUM/ MIN
      IREM = NUM - IG * MIN
      IF(IREM.EQ.0) GO TO 699
      NUM = MIN
      MIN = IREM
      GO TO 710
  699 CONTINUE
      L = 0
C DETERMINATION OF 'B' ARRAY
C DETERMINIATION OF MOD VALUES
      DO 11 I = 1, N
      A = NR(I)
      B = MIN
      C = A/B
C 'IG' IS THE GREATEST INTEGER
      IG = C
      IF(C-IG) 11, 11, 12
C 'L' IS THE COUNTER OF INTEGERS NOT EVENLY DIVISIBLE BY
C THE MINIMUM
```

```
   12 L = L +1
      MINIG = MIN * IG
C 'IB(I)' IS AN ARRAY OF B VALUES
      IB(L) = NR(I) - MINIG
      L = L + 1
      IB(L) = MIN + MINIG - NR(I)
   11 CONTINUE
      K = L
C 'M' IS THE SIZE OF THE 'B' ARRAY
      M = K + 1
      IF(L.NE.0) GO TO 16
      IGCD = MIN
      GO TO 50
   16 IB(M) = MIN - 1
C 'JF(I) IS THE ARRAY OF F VALUES
C NOTE PROGRAM LOGIC REQUIRES 'F'ARRAY INDICES
C INCREASED BY +1
C INITAIALIZING 'F' ARRAY TO MIN PLUS 1
      JF(1) = 0
      DO 13 I = 2, MIN
      JF(I) = MIN + 1
   13 CONTINUE
C 'ISTOP' RECORDS NUMBER OF CHANGES OF F VALUES
C IN THE PRESENT ITERATION
C 'KONT' IS AN ITERATION COUNTER
   29 ISTOP = 0
      KONT = KONT + 1
C DETERMINATION OF NEW F VAUES
      DO 14 I = 2, MIN
      IF(JF(I).EQ. 0) GO TO 14
      DO 15 J = 1, K
      IF (I - 1 - IB(J)) 21, 22, 23
C 'JJ' IS THE INDEX OF F
   21 JJ = I - IB(J) + MIN
      GO TO 24
   22 JJ = 1
      GO TO 24
   23 JJ = I - IB(J)
C DETERMINATION OF MINIMUM VALUES WHICH ARE NEW F VALUES
   24 IF ( JF(I) .LE. JF(JJ)) GO TO 15
      JF (I) = JF(JJ)
      ISTOP = ISTOP + 1
   15 CONTINUE
      IF(I-1-IB(M)) 25, 26, 27
   25 JJ = I- IB(M) + MIN
      GO TO 28
   26 JJ = 1
      GO TO 28
   27 JJ = I - IB(M)
   28 IF(JF(I) .LE. JF(JJ) + 1) GO TO 14
      JF(I) = JF(JJ) + 1
      ISTOP = ISTOP + 1
   14 CONTINUE
      IF(ISTOP .GT. 0) GO TO 29
```

```
C NO CHANGED F VALUES INDICATES PROCEDURE COMPLETE
C 'IGCD' IS THE GREATEST COMMON DIVISOR

      IGCD = JF(2) + 1

C END OF EXECUTION TIMING

   50 CONTINUE
      WRITE (6, 100)
  100 FORMAT(1H1,45X,'THE GREATEST COMMON DIVISOR OF ;'
     1//)
      WRITE (6,101) (NR(I) , I = 1,N)
  101 FORMAT(20X,10I8,//)
      WRITE (6,102) IGCD
  102 FORMAT(/////,55X,'IS ', I4)
      WRITE(6,103) MIN,M, KONT
  103 FORMAT(//,32X,'THE F MATRIX IS ',I5,' BY ',I5,'.',
     15X,I4,'  ITERATIONS USED')
      WRITE (6,502) N, IFUD, IR1, IR2
  502 FORMAT(//,32X,I5,' NUMBERS, MULTIPLES OF',I3,
     1' OVER THE RANGE',I4, ' -',I5)
      IX = KMA / 3
      IF(KNT.NE.3) GO TO 750
  600 CONTINUE
      END
```

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	20
2.	Library Naval Postgraduate School Monterey, California 93940	2
3.	Director, Systems Analysis Division (OP 96) Office of the Chief of Naval Operations Washington, D. C. 20350	1
4.	Department of the Army Civil Schools Branch, OPO, OPD Washington, D. C. 20315	2
5.	Professor Harold Greenberg Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	1
6.	Major Harry G. MacGregor 702 Fourth Avenue Fort Ord, California 93941	1
7.	Captain Kent A. Modine 311 Hayes Circle Fort Ord, California 93941	1
8.	Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	1

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D			
(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)			
1. ORIGINATING ACTIVITY *(Corporate author)* Naval Postgraduate School Monterey, California 93940		2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP	
3. REPORT TITLE Methods for Computing the Greatest Common Divisor and Applications in Mathematical Programming			
4. DESCRIPTIVE NOTES *(Type of report and, inclusive dates)* Thesis			
5. AUTHOR(S) *(First name, middle initial, last name)* Harry G. MacGregor, Jr. Kent A. Modine			
6. REPORT DATE June 1968		7a. TOTAL NO. OF PAGES 46	7b. NO. OF REFS 7
8a. CONTRACT OR GRANT NO. b. PROJECT NO. c. d.		9a. ORIGINATOR'S REPORT NUMBER(S) 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*	
10. DISTRIBUTION STATEMENT [redacted]			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY Naval Postgraduate School Monterey, California 93940	

13. ABSTRACT

Several methods are presented for determining the greatest common divisor of a set of positive integers by solving the integer program: find the integers x_i that minimize $Z = \sum_{i=1}^{n} a_i x_i$ subject to $Z \geq 1$. The methods are programmed for use on a computer and compared with the Euclidean algorithm. Computational results and applications are given.

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Greatest Common Divisor						
Integer Programming						