Downham/ done for SP

1, p. 121.

common χ^2 tests, *Biometrics*, Vol. 10, p. 417.
tive congruential pseudo-random number generators, *Comp. J.*,

Roy. Met. Soc., Vol. 26, p. 250.
ber generating sequences for digital computers, Office of Technical

andom number generators for binary machines, *JACM*, Vol. 11,

domness in a series of numerical observations, *Proc. Roy. Soc. of*

of numbers, *Proc. Roy. Soc. of Edin.*, Vol. 57, p. 332.
ator, *CACM*, Vol. 8, p. 350.
of runs up and down, *Ann. Math. Stat.*, Vol. XV, p. 58.
ial pseudo-random number generators, *Comp. J.*, Vol. 11, p. 341.
ator, *JACM*, Vol. 7, p. 75.

I think his numbers are different by 1 from mine.

Review

space vehicle. Whether such tactics will ever be economic for terrestrial communication is another matter. But the mathematical interest is great and involves very varied topics. For example, the code discussed in Chapter 2 is presented graphically as a rectangular matrix of black and white squares: this is particularly appropriate since its construction is based on the Hadamard matrix which was first described in connection with the design of tessellated pavements.* The 'Fast Fourier Transform', which is a product of computer programming, has proved invaluable in the de-coding of certain types of code. The construction of error correcting codes may be based on combinatorial algebra or on topology. If some British mathematicians can be persuaded to read this book, and if as a result they become interested in the mathematical problems of error correcting codes, this may help to bring Britain forward in a subject in which we lag sadly behind the space-inspired Americans and Russians.

D. A. BELL (Hull)

*SYLVESTER, J. J., 'Thoughts on inverse orthogonal matrices, simultaneous sign successions, and tessellated pavements in two or more colours, with applications to Newton's rule, ornamental tile-work and the theory or numbers.' *Phil. Mag.* (ser.4), 34, 461–475 (1867).

Computer Journal

LUNNON → A/212

A postage stamp problem

By W. F. Lunnon*

The postage stamp problem consists of choosing, for a given n and m, a set of n integers such that
(*a*) sums of m (or fewer) of these integers can realise the numbers 1, 2, 3, . . . , $N-1$,
(*b*) the value of N in (*a*) above is as large as possible.
This paper discusses a computational approach to the problem.

(Received February 1969)

What is the next number in the sequence 23456789...? See **Table 1** for $n = 3$, second column. Hard luck. The answer is nine.

To comprehend—or at least assimilate—this gross inductive anomaly it is necessary to familiarise oneself with the problems of a Post Office in a distant land. This body issues stamps in sets of n denominations and forbids customers to stick more than m on one letter. They wish to choose, for given n and m, a set which will give the greatest consecutive range of postal rates from one cent upwards. For example, if $n = 4$ and $m = 3$, the best set is (1, 4, 7, 8) cents; with up to 3 of these stamps any rate less than 25 cents may be obtained.

Formally, let n, m be numbers (i.e. non-negative integers). Let $S = S(1), S(2), \ldots, S(n)$ be a set of distinct numbers (stamp denominations) such that $S(1) < S(2) < \ldots < S(n)$. r (a postal rate) is 'obtainable' from S if there exists another set T (a choice of stamps) such that

$$r = \sum_{i=1}^{n} T(i)S(i) \quad \text{and} \quad \sum_{i=1}^{n} T(i) \leqslant m. \qquad (1)$$

The 'value' $val(S, n, m)$, or just $val(S)$, of S is the least number not obtainable from S. A 'best set' (of which there may be more than one) for given n, m is one whose value is the maximum, $V(n, m)$.

E.g., suppose $n = 4$ and $m = 3$. If $S = (1\ 4, 7, 8)$ then 22 is obtainable from S since

$$22 = 0\times1 + 0\times4 + 2\times7 + 1\times8$$

and

$$0 + 0 + 2 + 1 \leqslant 3.$$

25 is the least number not so obtainable, so $val(S) = 25$. S is best since all other sets have lower values: e.g., $val(1, 4, 5, 16) = 18$.

In this paper we investigate the best sets (Table 1) and their values $V(n, m)$ (**Table 2**). We must confess to being interested more in the computation than in the possibility of a theoretical solution: in fact, the appearance of the latter would now be a positive embarrassment.

Study of the tables suggests no imminent danger. They display irritating features: witness the perverse behaviour of $S(2)$ when $n = 3$, noted above, which caused the author to cry Eureka for a happy hour pending computation of $(n = 3, m = 9)$. Still keeping $n = 3$, observe how the best set for $m = 10$, (1, 10, 26), is also a best set for $m + 1 = 11$: for $m < 36$ this happens at $m = 10, 14, 17, 20, 24, 27, 29, 33$ (the rest of which do not appear in Table 2). When $m = 2$ and $n = 10$ or 11, the first two best sets differ only in $S(3)$ being 3 in the first and 5 in the second. For fixed m, as one might expect, a best set is often a subset of another for larger n: but more often it isn't. We now review the efforts made to explain these phenomena.

History

The earliest references to the problem seem to be in Sprague (1960) and Legard (1962). These ask for the minimum n for which some set exists with value greater than 100, where $m = 2$ and 3 respectively. If $m = 2$ one can construct by hand such a set with $n = 16$ stamps; to show whether this is minimal might involve computing $V(n, 2)$ up to $n = 15$, which is currently beyond us. If $m = 3$ a set with $n = 9$ can be constructed and, since $V(8, 3) = 94 < 100$, this is minimal.

Complete solutions are trivial for $(n = 1$, any $m)$ and (any n, $m = 1$). For $(n = 2$, any $m)$ it is easy to prove that the best sets are—taking nearest integers—$(1, \frac{1}{2}(m + 3))$, with value $(\frac{1}{2}(m + 3))^2 - 1$.

Several people have worked on the further computational and theoretical aspects, mainly at Cambridge University with the inspiration of J. C. P. Miller and without, unfortunately, publication. Among these J. A. Grant computed $V(8, 3)$ and B. Landy $V(9, 3)$ around 1963 (since confirmed by M. L. V. Pitteway): since our own program would take 40 hours to repeat the feat, we have used Landy's value for $V(9, 3)$ in the tables. Most of the rest of our results are confirmed independently by M. F. Challis.

A. Henrici (and independently Hofmeister, 1968) has investigated $(n = 3$, any $m)$ and is said to have a complete solution. For this reason, although we have not seen his work, we have not extended our table for $n = 3$ any further. Henrici has also looked at (any n, $m = 2$); the solution of an altered form of this case, where the best set for n is required to be a subset of the best set for $n + 1$, would for example show how dense an infinite set must be to satisfy Goldbach's hypothesis.

Some attention has been paid to symmetric solutions, where $S(i)$ is in the set iff $S(n) - S(i)$ is. Wegner and Doig (1966) investigate sets called 'j-bases' which possess a nested extension of this property, and R. D. Wycherley has computed some best j-bases for $m \leqslant 17$. We have

* *Department of Computer Science, University of Manchester*
Now at *SRC Atlas Computer Laboratory, Chilton, Didcot, Berkshire*

p. 377

mpared them with our own, except to note that $m = 2$ they are best if $n \leq 8$ and quite good thereafter.

Programming aspects

It is convenient to think of the set S as augmented by an extra stamp $S(0) = 0$ and insisting that exactly m stamps are used per letter.

We require some bound on the sizes of the $S(i)$. Obviously

$$S(i-1) < S(i) \leq val(S(0), \ldots, S(i-1)). \quad (2)$$

The *val* on the right must be at most the number $U(n, i)$ of ways to choose with repeats i things from $n+1$, so

$$S(i) \leq V(n, i) \leq U(n, i) = \binom{n+i}{n}. \quad (3)$$

Incidentally, $V(n, m)/U(n, m)$ measures the 'efficiency' of the best set, and for this reason we have included both functions in Table 2. The ratio might prove interesting.

Another bound which would be reassuring is on the number of best sets for given n and m. We assume that there are fewer than $n + m$, and have prepared ourselves gallantly to sacrifice the excess should the assumption fail. The record so far is for $m = 2$, $n = 6$, with 5 best sets.

For given n, m the 'serial method' of finding best sets breaks up into two computational parts: to enumerate all sets S within some plausible universe, where the object is to enumerate as few as possible; and, given a set S to find its value, where the object is to compute this as quickly as possible.

Alternatively, we might commence with an entire class S' of plausible sets S and, for each rate r in turn, eliminate those S from which r is unobtainable. When S' vanishes, $r = V(n, m)$. The storage problem is insuperable, even if S' is stored as a tree: however, a mixture of the two approaches might be effective. For example, once $S(1), \ldots, S(n-1)$ have been chosen serially we could generate the set of all $S(n)$ which yield a value as good as the current best: this set would usually be empty. Generating it efficiently invites ingenuity, but we feel that any resulting improvement could at best add one more hyperbola on the right of the tables; and so we return to the purely serial method.

Table 1

Best sets for n stamps in set, m on letter

m/n	1	2	3	4	5	6	7	8	9
1	1	1 2	1 2 3	1 2 3 4	1 2 3 4 5	1 2 3 4 5 6	1 2 3 4 5 6 7	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8 9
2	1	1 2 1 3	1 3 4	1 3 5 6	1 3 5 7 8	1 2 5 8 9 10 1 3 4 8 9 11 1 3 4 9 11 16 1 3 5 6 13 14 1 3 5 7 9 10	1 2 5 8 11 12 13 1 3 4 9 10 12 13 1 3 5 7 8 17 18	1 2 5 8 11 14 15 16 1 3 5 7 9 10 21 22	1 3 4 9 11 16 17 19 20
3	1	1 3	1 4 5	1 4 7 8	1 4 6 14 15	1 3 7 9 19 24 1 4 6 14 17 29	1 4 5 15 18 27 34	1 3 6 10 24 26 39 41	1 3 8 9 14 32 36 51
4	1	1 3 1 4	1 5 8	1 3 11 18	1 3 11 15 32	1 4 9 16 38 49 1 5 8 27 29 44	1 4 9 24 35 49 51 1 4 10 15 37 50 71 1 5 8 25 31 52 71		
5	1	1 4	1 6 7	1 4 12 21 1 5 12 28	1 4 9 31 51	1 4 13 24 56 61 1 5 8 33 54 67			
6	1	1 4 1 5	1 7 12	1 4 19 33	1 7 12 43 52	1 7 11 48 83 115			
7	1	1 5	1 8 13	1 5 24 37	1 8 11 64 102				
8	1	1 5 1 6	1 9 14	1 6 25 65	1 9 15 78 115 1 9 15 80 118				
9	1	1 6	1 9 20	1 5 34 60	1 9 23 108 181				
10	1	1 6 1 7	1 10 26	1 6 41 67					
11	1	1 7	1 9 30 1 10 26	1 7 48 85					
12	1	1 7 1 8	1 11 37	1 7 48 126					
13	1	1 8	1 13 34	1 9 56 155					
14	1	1 8 1 9	1 12 52	1 8 61 164					

m/n	10	11	12
1	1 2 3 4 5 6 7 8 9 10	1 2 3 4 5 6 7 8 9 10 11	1 2 3 4 5 6 7 8 9 10 11 12
2	1 2 3 7 11 15 19 21 22 24 1 2 5 7 11 15 19 21 22 24	1 2 3 7 11 15 19 23 25 26 28 1 2 5 7 11 15 19 23 25 26 28 1 3 4 9 11 16 18 23 24 26 27 1 3 5 6 13 14 21 22 24 26 27	1 3 4 9 11 16 21 23 28 29 31 32

A posta

Enumeration

Using (2), and assuming we have an **integer procedur** *val*(*S*, *n*, *m*) on hand, this looks like

```
S[0] := 0;
for S[1] := val(S, 0, m) step −1 until S[0] + 1 do
for S[2] := val(S, 1, m) step −1 until S[1] + 1 do
        .
        .
        .
for S[n] := val(S, n − 1, m) step −1 until
    S[n − 1] + 1 do
        [test val(S, n, m) for best so far];
```

This sort of indefinitely nested **for** loop occurs frequently in combinatorial problems, under some such name as 'back-tracking' (Beckenbach, 1964, Floyd, 1967, and Walker, 1960). It may be thought of as a **for** loop in which the index variable is a vector *S*[0:*n*] instead of the usual scalar. To find the next *S* after a given one, the idea is to back down from *S*[*n*] till an *S*[*i*] is found which has not reached its limit, then to increment this *S*[*i*] and re-initialise all subsequent elements of *S*. For example the present instance is actually coded as follows:

T

Best values $V(n, m)$ and number of choi

m/n	1	2	3	4	5
1	2	3	4	5	6
	2	3	4	5	6
2	3	5	9	13	17
	3	6	10	15	21
3	4	8	16	25	37
	4	10	20	35	56
4	5	11	27	45	71
	5	15	35	70	126
5	6	15	36	72	127
	6	21	56	126	252
6	7	19	53	115	217
	7	28	84	210	462
7	8	24	70	166	346
	8	36	120	330	792
8	9	29	90	235	513
	9	45	165	495	1287
9	10	35	113	327	798
	10	55	220	715	2002
10	11	41	147	428	
	11	66	286	1001	
11	12	48	173	548	
	12	78	364	1365	
12	13	55	213	709	
	13	91	455	1820	
13	14	63	260	874	
	14	105	560	2380	
14	15	71	303	1095	
	15	120	680	3060	

re are r than $n + m$, and have prepared ourselves
lantly to sacrifice the excess should the assumption
. The record so far is for $m = 2$, $n = 6$, with 5 best
.

For given n, m the 'serial method' of finding best sets
aks up into two computational parts: to enumerate
sets S within some plausible universe, where the
ect is to enumerate as few as possible; and, given a
S to find its value, where the object is to compute this
quickly as possible.

lternatively, we might commence with an entire
ss S' of plausible sets S and, for each rate r in turn,
ninate those S from which r is unobtainable. When
vanishes, $r = V(n, m)$. The storage problem is
uperable, even if S' is stored as a tree: however, a
xture of the two approaches might be effective. For
ample, once $S(1), \ldots, S(n-1)$ have been chosen
ially we could generate the set of all $S(n)$ which yield
alue as good as the current best: this set would usually
empty. Generating it efficiently invites ingenuity,
t we feel that any resulting improvement could at best
d one more hyperbola on the right of the tables; and
we return to the purely serial method.

set, *m* on letter

8 9

3 4 5 6 7 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 9

5 8 11 12 13 1 2 5 8 11 14 15 16 1 3 4 9 11 16 17 19 20
4 9 10 12 13 1 3 5 7 9 10 21 22
5 7 8 17 18

5 15 1 1 3 6 10 24 26 39 41 1 3 8 9 14 32 36 51 53

9 24 35 49 51
10 15 37 50 71
8 25 31 52 71

3 4 5 6 7 8 9 10 11 12

4 9 11 16 21 23 28 29 31 32

numeration

Using (2), and assuming we have an **integer procedure** *val*(S, n, m) on hand, this looks like

```
S[0] := 0;
for S[1] := val(S, 0, m) step -1 until S[0] + 1 do
for S[2] := val(S, 1, m) step -1 until S[1] + 1 do
    .
    .
    .
for S[n] := val(S, n - 1, m) step -1 until
      S[n - 1] + 1 do
            [test val(S, n, m) for best so far];
```

This sort of indefinitely nested **for** loop occurs frequently in combinatorial probiems, under some such ame as 'back-tracking' (Beckenbach, 1964, Floyd, 1967, nd Walker, 1960). It may be thought of as a **for** loop n which the index variable is a vector $S[0:n]$ instead of he usual scalar. To find the next S after a given one, he idea is to back down from $S[n]$ till an $S[i]$ is found vhich has not reached its limit, then to increment this $S[i]$ and re-initialise all subsequent elements of S. For xample the present instance is actually coded as follows:

```
    i := 0; S[i] := 1;
L:      S[i] := S[i] - 1;
    for i := i + 1 step 1 until n do
            S[i] := val(S, i - 1, m);
    [test val(S, n, m) for best so far];
    for i := n step -1 until 1 do
            if S[i] ≠ S[i - 1] + 1 then goto L;
```

One attempt to further reduce the enumeration (due to J. S. Rohl) reasons that if v is the best value found so far, there is no point in considering sets for which $S(n) < (v-1)/m$ or $S(n-1) < (v-1)/m^2$, etc. Using this device it seems natural to enumerate the $S(i)$ downwards (as is done in the coding above) to avoid calling *val* unnecessarily. However, this may be a mistake if the best sets turn out to be clustered near the end of the enumeration. Our program incorporates the idea in this form, but we have not established its utility nor answered the last query.

It is possible to compute simultaneously with $V(n, m)$ all $V(i, j)$ for $i, j \leqslant n, m$, at some cost in speed. Owing to the severely exponential increase in time with m and especially n, this is only worthwhile for $n = 4$ (time $= O(2^m)$) and $n = 5$ (time $= O(4^m)$). The point

Table 2

Best values $V(n, m)$ and number of choices $U(n, m)$ for n stamps in set, m on letter

m/n	1	2	3	4	5	6	7	8	9	10	11	12
1	2 2	3 3	4 4	5 5	6 6	7 7	8 8	9 9	10 10	11 11	12 12	13 13
2	3 3	5 6	9 10	13 15	17 21	21 28	27 36	33 45	41 55	47 66	55 78	65 91
3	4 4	8 10	16 20	25 35	37 56	53 84	71 120	94 165	122 220			
4	5 5	11 15	27 35	45 70	71 126	109 210	163 330					
5	6 6	15 21	36 56	72 126	127 252	212 462						
6	7 7	19 28	53 84	115 210	217 462	389 924						
7	8 8	24 36	70 120	166 330	346 792							
8	9 9	29 45	90 165	235 495	513 1287							
9	10 10	35 55	113 220	327 715	798 2002							
10	11 11	41 66	147 286	428 1001								
11	12 12	48 78	173 364	548 1365								
12	13 13	55 91	213 455	709 1820								
13	14 14	63 105	260 560	874 2380								
14	15 15	71 120	303 680	1095 3060								

380

ntioning this here is that Rohl's device does not prevent it: no best sets for smaller n are lost because

$$V(n, m)/m^{n-i} \leqslant V(i, m),$$

and none for smaller m because

$$V(n, m)\, m^{n-i} \leqslant (m/j)V(n, j)/m^{n-i} \leqslant V(n, j)/j^{n-i}.$$

Valuation

Because of the way the *val* routine is used above, when it is called upon to value $S(1), \ldots, S(i)$ we may assume that $S(1), \ldots, S(i-1)$ are unchanged from last time: so the portion of the calculation relating to them need not be repeated.

Given a set S of stamps, we define $L(i, j)$ to be the set of all numbers obtainable from $S(1), \ldots, S(i)$ using up to j stamps. A convenient way to store L is as a string of bits: bit $k = 1$ iff k is obtainable. For example, if $S = (1, 4, 5)$ then

$$\left.\begin{array}{l} L(2, 3) = 11111\ 11011\ 00100\ 00000\ \ldots \\ L(3, 2) = \quad\ 11101\ 11011\ 10000\ \ldots \\ L(3, 3) = 11111\ 11111\ 11111\ 10000\ \ldots \end{array}\right\} \quad (4)$$

Observe that $L(3, 3)$ has a 1 in just those places where $L(2, 3)$ has or $L(3, 2)$ shifted up 5 places has: in general,

$$L(i, j) = L(i, j-1) \uparrow S(i) \vee L(i-1, j) \cdot \quad (5)$$

and furthermore

$$L(i, 0) = L(0, j) = 10000\ldots \quad (6)$$

since 0 is the only number obtainable in these cases.

Using (5) and (6) $L(i, m)$ can be constructed from $L(i, 0)$ and the $L(i-1, j)$, $1 \leqslant j < m$, all of which have already been calculated. Then $val(S, i, m)$ is just the position of the first zero in $L(i, m)$.

Details to watch: The two lists on the RHS of (5) may fail to overlap, i.e. $\max(L(i-1, j))$ may be less than i. This results in a gap which has to be coded for. The merge (5) may leave empty words on the end of the resulting $L(i, j)$, which should be pruned. In the inner loop $L(n, j)$ could be pruned even more by stopping at $mV(n-1, m) - (m-j)S(n)$, since $L(n, m)$ must have its first zero before $mV(n-1, m)$: this would be effective for high values of m. At some extra effort one could also drop the initial string of 1's and replace it by a base count. We didn't think of either device at the time.

Long strings of 1's and 0's suggest storing an L-list as bit-changes: e.g. $L(2, 3)$ above would be kept as (7, 8, 10, 12, 13), these being the places where 0 changes to 1 or 1 to 0. However, there is no reason to suppose that even good sets are not more or less random in the middle of their L-lists. The idea is feasible if the overall ratio changes/bits is less than 1/word length (24 bits on Atlas), assuming equal times for dealing with one word of list and one change. Even this may be over-optimistic, for the implementation of (5) becomes rather complicated.

Performance

It is the lot of every machine-code programmer to see his most cherished creations sink into unpublishable obscurity. Enough! let us baldly state that the value algorithm was hand-coded for the Manchester University Atlas I computer, the rest of the program being in Atlas Autocode, and Table 1 and Table 2 were produced in about 20 hours. The time required is badly exponential in n, slightly less so in m. The inner loop—merging the $L(i, j)$ together as in (5)—took about one microsecond (less than half an instruction) per bit. The results on the right-hand hyperbola of the tables each took several hours, the longest being $n = m^7 = 6$ with $5\frac{1}{2}$ hours during this time the value routine was called 4 million times, and the L-lists extended to 10 thousand bits. Run-times of this order demand a dump/restart facility in order not to upset either the programmer (when the machine stops) or the computing service (when it doesn't).

Acknowledgements

The author gratefully acknowledges the advice of C. F. J. Outred, and of J. C. P. Miller who supplied the historical information.

References

Beckenbach, E. F. (Ed.) (1964). *Applied Combinatorial Mathematics*. New York: John Wiley and Sons, Inc.
Floyd, R. W. (1967). Non-Deterministic Algorithms, *JACM*, Vol. 14, pp. 636–645.
Hofmeister, G. R. (1968). Doctoral Dissertation (unpublished), Mainz.
Legard, A. (1962). Brain Teaser, *Sunday Times*, 23 December 1962 and 20 January 1963.
Sprague, R. P. (1960). *Unterhaltsame Mathematik*. Braunschweig: Vieweg & Sohn. English translation by T. H. O'Be
(1963), *Recreations in Mathematics*. Blackie.
Walker, R. J. (1960). An Enumerative Technique for a Class of Combinatorial Problems, *Proc. Symp. App. Maths.*, Vol
pp. 91–94.
Wegner, P., and Doig, A. (1966). Symmetric Solutions of the Postage Stamp Problem. *Revue Française de Recherche Op*
tionelle, Vol. 41, pp. 353–374.

Initialising Geoffrion's implicit the zero-one linear programmin

By J. L. Byrne* and L. G. Proll†

This paper describes a me programs in zero-one variab which may be employed in between the original and mo

(Received December 1968,

1. Introduction

In a recent paper Geoffrion (1967) has proposed an implicit enumeration algorithm for the solution of the general linear programming problem in zero-one variables which is computationally attractive because of its simplicity and modest storage requirements compared with the related algorithms of Balas (1965) and Glover (1965). However, the computational experience with Geoffrion's algorithm provided by Freeman (1966) and Byrne (1967) shows that the time required to reach termination may be considerable even for relatively small problems. The aim of this paper is to show how this time may be considerably reduced by means of a modification of the non-iterative part of the algorithm. The modification proposed consists of three modules, one or all of which may be employed depending on the path along which the computation flows.

In the following sections the general linear programming problem in zero-one variables is referred to in the following form:

$$\begin{aligned} &\text{minimise } z = c'.x \\ &\text{subject to } Ax \geqslant b, \qquad (1) \\ &\qquad x_j = 0 \text{ or } 1, (j = 1, 2, \ldots, n), \\ &\qquad c \geqslant 0, \end{aligned}$$

where c, x are n-vectors, b is an m-vector and A is an m by n matrix. The coefficients in (1) are not restricted to being integers. Any linear programming problem in zero-one variables can be written in the form of (1) by means of a series of simple transformations (e.g. see Balas, 1965). The above form differs slightly from that considered by Geoffrion in the expression of the constraints but appears more natural to the authors.

2. Some aspects of Geoffrion's algorithm

In this section some aspects of Geoffrion's algorithm which are necessary to the development of the proposed modification are reviewed.

A solution of (1) is any binary n-vector. A solution x of (1) is feasible if $Ax \geqslant b$ and a feasible solution which minimises z over the set of feasible solutions is said to be optimal. The objective of an implicit enumeration algorithm is to obtain and verify an optimal feasible solution whilst explicitly enumerating as few as possible of the 2^n solutions of (1). This latter point is an

* *Department of Mathematics, University of Queensland*
† *Department of Mathematics, University of Southampton*