NUMER ICAL EXPERIMENTS WITH THE SPECTRAL TEST

by

R. William Gosper

STAN-CS-75-490
MAY 1975

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERS ITY

Numerical Experiments With The Spectral Test

Abstract

Following Marsaglia and Dieter, the spectral test for linear congruential random number generators is developed from the grid or lattice point model rather than the Fourier transform model, Several modifications to the published algorithms were tried, One of these refinements, which uses results from lesser dimensions to compute higher dimensional ones, was found to decrease the computation time substantially. A change in the definition of the spectral test is proposed in the section entitled "A Question of Independence".

Background

The values of the LCRNG (Linear Congruentiat Random Number Generator)

$$x_{i+1} \equiv a x_i + c \mod m$$

when plotted on the x axis, lie on a one dimensional grid. That is, the difference between any pair of them is a multiple of some integer ∂:

$$x_j - x_i = L \partial$$

for some integer L. ∂ is 1 for a maximum period RNG, but is at least 2 if c = 0 and m is a power of 2, Depending upon the number theory underlying the choice of a, c, and m, it is possible that not all of the m/∂ grid points between 0 and m-1 will be generated,

If instead we plot consecutive pairs of values

$$(x_1, x_2), (x_2, x_3), \ldots, (x_i, x_{i+1}), \ldots$$

as (x, y) coordinates, we will get a two dimensional grid. This means that the (vector) difference between any pair of points is the sum of integer multiples of two constant vectors, called basis vectors, which define the grid. To find these basis vectors, we have

$$X_{i+1} = a x_i + c - K m$$

$$x_{j+1} = a x_j + c - M m$$

for some integers K and M, and so

$$(x_j, x_{j+1}) - (x_i, x_{i+1}) = (L \partial, a L \partial - (M-K) m)$$

$$= L \partial (1, a) - (M-K) (0, m)$$

This research was supported in part by National Science Foundation grant GJ 36473X and by the Office of Naval Research contract NR 044-402 Some computer experiments reported herein were done via the ARPA network with the MACSYMA system, a project supported by the Advanced Research Projects Agency under ONR contract N00014-70-A-0362-0001. Reproduction in whole or in part is permitted for any purpose of the United States Government.

a sum of integer multiples of the basis vectors

$$\partial\ (1,\ a)\ \text{and}\ (0,\ m)\ .$$

Similarly, if consecutive triplets are plotted in 3 space, we will find a three dimensional grid with basis

$$\partial\ (1,\ a,\ a^2),\ (0,\ m,\ 0),\ \text{and}\ (0,0,\ m)\ .$$

This pattern extends to n dimensions. We will call such a grid basis F and denote its vectors by f_j, $1 \le j \le n$,

We can regard an n dimensional grid as an x dimensional grid of n - x dimensional grids and say that x of the n basis vectors "connect" identical copies of the n - x dimensional subgrid generated by the remaining n - x basis Vectors. This even works when x = n, if we define a zero dimensional grid to be a point. (Try rereading this with n = 2 or 3 and x = 1 or 2.)

Note that the only way that these vectors, and thus the grid's structure, depend on the "increment" c is in the value of ∂, the smallest difference between RNG outputs. If c is chosen so that ∂ is greater than 1, we will only generate every ∂th subgrid on the one dimensional grid generated by

$$(1,\ a,\ a^2,\ a^3,\ \ldots)\ .$$

The only other effect of varying c is to shift the entire grid and to change the order in which the grid points are generated.

There are infinitely many bases for a given grid, for if cl and c are two basis vectors in an n dimensional grid, then we can replace d by d - c and still reach the same points, since we can always rewrite

$$K\,d + L\,c \quad \text{as} \quad K\,(d - c) + (K+L)\,c\ .$$

We could repeat this process q times to replace d by d - q c, and we could also involve other pairs of vectors, In the particular case of the basis

$$\partial\ (1,\ a,\ a^2),\ (0,\ m,\ 0),\ \text{and}\ (0,0,m)$$

we could reduce the powers of a modulo m by subtracting appropriate multiples of the latter two vectors from the first.

Now suppose, for example, that we are examining the three dimensional grid of some LCRNG. It will lie entirely within the cubical region

$$0 \le x < m\ ,\ 0 \le y < m\ ,\ 0 \le z < m$$

since all of the RNG values are modulo m.

We could imagine as a physical model a clear plastic cube containing m/d dots (in one to one correspondence with the one dimensional grid). As we rotate the cube in various ways, we uii I see the dots arranged in various planar grids, corresponding to various Choices of bases.

If we can orient this cube so that all of the points fall in a few widely separated planes, we should be dissatisfied with the RNG that put them there, since the gaps between these planes represent large regions of impossible triplets of consecutive outputs, Concomitantly, these few planes would be undesirably crowded with the points that should be occupying the intervening gaps. Thus, the widest separation between n-1 dimensional subgrids is a measure of the uniformity of the RNG, when its outputs are grouped n at a time,

The determination of this distance is the spectral test in n dimensions.

If the grid were a regular cubic one, there would be a basis of three mutually perpendicular vectors of equal length. Since the whole grid would contain m/d points, there would be about

$$(m/d)^{1/3}$$

points along each one dimensional subgrid, making the basis vectors about

$$d^{1/3}\ m^{2/3}$$

in length. Unfortunately the (unreduced) basis vectors defined by the LCRNG are anything but short and regular. In fact, they are so long

that they all protrude from the cube of interest! (Assuming $a^2 > m$ as Knuth et al recommend.)

In order- to resolve such a grid into more widely separated, more dense lypopulated subgrids, we must find shorter and more nearly perpendicular basis vectors. We can do this with an algorithm analogous to Euclid's GCD--by replacing a given basis vector with the "remainder" resulting from subtracting some other basis vector from it as many times as will minimize the remainder's length, By analogy with ordinary division, this number of possible subtractions can be called the "quotient" q, and can be computed directly instead of by tedious iteration:

$$cl = \text{round}\left(\frac{c \cdot cl}{d \cdot d}\right)$$

where cl is the divisor, c is the dividend, and c - q d is the remainder. cl is the component of the dividend par-allel to the divisor, divided by the length of the divisor, then finally rounded to the nearest integer.

This will leave the remainder with the least possible component parallel to the divisor, subject to the quotient being an integer to preserve the grid,

Tradition suggests that once a divisor is chosen, it should be remaindered with all of the other basis vectors.

Eventually this process, call it the F process, will get stuck when all quotients of pairs are zero. It would be nice if, after all this work, we could be sure that the resulting basis contained the vector linking the most widely separated subgrids. Unfortunately, example 8 will show this hope to be vain. We have, however, two more weapons. The neater of them is another collection of n vectors, computed from the basis, and called the dual basis. We will denote it by E and its elements by e. They have the following magic property:

for $1 \le i, j \le n$,

$$e_i \cdot f_j = 0 \quad \text{if } i \neq j, \text{ but}$$

$$e_i \cdot f_i = 1.$$

This means that the jth dual basis vector is perpendicular to the n-1 dimensional subgrid copies connected by the jth vector of the original basis, and conveniently, the length of the dual vector is just the reciprocal of the perpendicular separation between these copies. Thus, to perform the spectral test, we need merely find the shortest vector in the dual grid. (Technically, there are many dual grids corresponding to shifting all of the points by any constant, but the vectors between them, and hence the dual basis, remain the same.)

First we find the dual basis from its definition, by forming an n by n matrix, f, whose rows are the original basis vectors. Then we invert and transpose it to form the dual matrix e. Repeating this one gives f again, verifying the duality. Since subtracting the ith vector from the jth in one basis corresponds to adding the jth to the ith in the other basis, we will still have a basis if we run the F process on the dual basis instead of on the original one. Call this the E process. The recipe for the spectral test now might read: Compute the dual basis from the original grid basis, run the E process until all the dual vectors are of minimal length, and return the reciprocal of the shortest. Unfortunately, there is no way to guarantee that the vectors are minimal when the E process gets stuck (example 2, n = 4).

As Knuth has observed (3.3.4, ex. 22, 23), a good strategy is to maintain both bases, and switch from E to F when E gets stuck, simply by interchanging the two matrices with respect to the remaindering operation. Then, if F gets anywhere, return to the E process. When F gets stuck, the algorithm quits, even though F's dying attempts may have unstuck E. Sad to say, after all of this there is still a slim chance that the shortest vector left in the dual basis is not the shortest expressible as a sum of integer multiples of dual vectors (although the author has never seen this except in cases where the matrices were first transformed by an experimental process which was neither E no F. Remember to see example 8.) Thus we resort to the ultimate weapon: exhaustive search.

Fortunately, Coveyou and MacPherson have shown that if an integer combination of the vectors $e_j, 1 \le j \le n$, is to be of minimal length, the coefficient of e_j cannot exceed

$$c_j = \lfloor |e_{\min}| / |f_j| \rfloor$$

in magnitude, where $e_{\min}$ is the shortest e_i and f_j is the jth vector of the dual basis (in this case the dual of the dual, i.e. the current basis for the LCRNG grid). Thus if e is the square matrix with basis vectors as rows, we must minimize the length of the vector ze over all nonzero vectors of integers z with

$$|z_j| \le |c_j| .$$

We can omit those z which are merely the negatives of ones already tried, for a total search of

$$\frac{(2c_1 + 1)(2c_2 + 1) \ldots (2c_n + 1) - 1}{2}$$

cases. Fortunately, only one c_j has ever been as large as 2 in the author's experience with the combined E and F strategy, (We have been using stuck to mean n consecutive failures to reduce the search volume, rather than n - 1, hence n, consecutive divisors with all quotients 0.)

The process described above can be refined in several ways.

Refinement 0

Scale up E, the dual basis, by a factor of m to avoid fractional elements. In fact, most derivations of the spectral test regard this integral dual basis as an automatic consequence of scaling the grid basis clown by a factor of m to fit in the unit cube.

Refinenient 1

One step of the E process, using e_j as divisor, can affect (shorten) only those e_i for which q_i is $\neq 0$. In the F basis, only f_j can change, very occasionally growing longer (example 1), (thus increasing c and thus the search length). Similarly, one step of the F process will only affect certain f_i, and only one e_j.

A clever algorithm could save itself many square roots by minding this.

For typical spectral tests (with $n > 2$), the E process produces much more rapidly convergent c than the F process, even though the E process will occasionally lengthen an f_j. A good algorithm might retract any step which increased the exhaustive search volume, or perhaps save up and return to the best bases if the last few steps lengthen the exhaustive search just before getting stuck.

It appears, however, that at least for LCRNG grids, search volumes hardly ever grow significantly. Thus, a version of the program was modified to run each process until all of the quotients in its matrix were 0, and only then compute the c vector. This seemed to gain about 20% over Dieter's strategy of defining a process to be stuck after n consecutive failures to beat the previous smallest search volunie.

Refinement 2

∂ must always divide m, and thus the entire grid **basis,** Therefore the entire grid may, in effect, be scaled down by ∂ if we take m t 0 mean modulus/∂, instead of just the modulus,

Thus, the basis matrices are initialized:

f, the grid matrix

$$\begin{pmatrix} 1 & a & a^2 & \cdots & a^{n-1} \\ 0 & m & 0 & \cdots & 0 \\ 0 & 0 & m & & 0 \\ \vdots & & & \ddots & \\ 0 & 0 & 0 & \cdots & m \end{pmatrix}$$

e, m times the dual matrix

$$\begin{pmatrix} m & 0 & 0 & \cdots & 0 \\ -a & 1 & 0 & \cdots & 0 \\ -a^2 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \\ -a^{n-1} & 0 & 0 & \cdots & 1 \end{pmatrix}$$

where, for efficiency, the powers of a are taken mod m.

Refinement 3

Often, the spectral test is desired for several consecutive n, in which case there is the opportunity to use the considerably reduced bases left over from the calculation in dimension n-1 to initialize the bases in dimension n. In fact, this technique is so successful that it is usually best to proceed incrementally from dimension 2, even if only one c(n) is desired, (Omitting the irrelevant exhaustive searches.) The crossover point is between n=3 and n-4, with a speedup factor of 2.5 typical for n=8.

The construction of the n dimensional bases from the previous ones is: adjoin to the right of f a new column which is $(a^{n-1} \bmod m)$ times its leftmost column, then adjoin the row

$$f_n = | 0\ 0\ 0\ \ldots\ m |$$

to the bottom, say, of f, For e, adjoin to the bottom the row

$$| -a^{n-1} \quad 0 \quad 0 \quad \ldots \quad 0 | \quad \bmod m$$

then to the right adjoin the column of all 0s and a 1.

A strange fact, illustrated by example 2, is that in order to profit from this incremental stratagem, one must interchange the processes so that F is used in preference to E. It is also important to use the new vector, f as the first divisor.
n

All this is because n -1 very large numbers have been introduced into the last column of f, and remaindering them by the n th vector reduces them mod m while any other operation on f or e just sort of spreads these big numbers around, Note that this incremental approach is not equivalent to starting with an n by n matrix and reducing just the first vector pair, then the first three, etc., since the F process would be stymied with only 0s to use against the higher powers of a, while the E process would be erroneously discouraged by rapidly growing c from those huge
j

number s accumulating in the rightmost columns of f.

After several hundred experiments, the author has only once seen this incremental method fail to find the minimal vector before the exhaustive search, This was also the only time a final c exceeded 1. (Example 8.)
j

Refinement 4

In the two dimensional case, the E and F processes are equivalent, since the e and f matrices can be interchanged by negating a row and column of each, and then swapping rows. These operations, if performed simultaneously on each matrix, preserve duality and cannot change anything about the shortest vector algorithms except possibly the order in which divisors are investigated, But in two dimensions there is only one possible first divisor, after which the divisors must alternate anyway,

Thus, when n = 2, it is unnecessary to switch processes, or even maintain the dual matrix and c , at least until the exhaustive search,

Even further simplification results from the observation that the two dimensional spectral test on a and m will emulate Bradley's refinement of Algorithm X (Knuth, 4.5.2) for the gcd of a and m, -except that it will stop about half way through.

Relation to the Serial Test

Both the spectral and serial tests investigate the uniformity of a whole period of n-tuples in an n dimensional cube, But the serial test merely measures the density of points in subcubes, while the spectral test, using the grid information, effectively tranforms these subcubes into a worst case orientation, so that, up to size

$$\frac{1}{|e|_{min}}$$

at least half would be empty and half would be proportionately too full. It seems paradoxical then, that the spectral test on

$2^{23} + 2^{12} + 5 \bmod 2^{35}$ could have the poor figure of merit .015 in two climensions and the very good figure 2.78 in three, since a generator flunking a low dimensional serial test should surely flunk a higher one. (See figure of merit definition in Explanation of Example Printouts section.1 The explanation is that, as n increases the number of points in the cube remains m/∂, and thus the distances' betcreen the nearest points grow, roughly as

$$\partial^{1/n}\, m^{1 - 1/n}.$$

Thus, if we are to compare serial tests in several different dimensions, we must increase the size of the subcuhes in such a way as to preserve their total number, else the point density counts will become unreliably small.

Now suppose we have an LCRNG whose n dimensional maximum subgrid separation is not much greater than for n - 1 dimensions. (Occasionally they can even be equal; see examples 2 and 3, n = 4 thru 8.1 Suppose further that we are serially testinq this RNG in n - 1 dimensions with a subcube edge length only slightly smaller than the grid separation, and that the subgrids are roughly parallel to the faces of the subcubes. Then many subcubes will fit or nearly fit between subgrids, causing severe density fluctuations, But when we jump to n dimensions, the increase in subcube size will mean that subcubes will no longer fit between the subgrids, thus drastically improving the serial test result.

A Question of Independence

One subtle difference between current formulations of the **serial and** spectral tests is that the serial test is performed on disjoint coordinate tuples, e g

$$(x_1, x_2), (x_3, x_4), \ldots$$

while the spectral test is performed on overlapping ones, eg

$$(x_1, x_2), (x_2, x_3), \ldots$$

The former sequence is clearly preferable, since it is unbiased.

To modify the spectral test to use disjoint tuples, we need merely determine the ∂ of the one dimensional grid resulting from using only every nth RNG value.

For instance, in a full period LCRNG, $\partial = 1$, but if the modulus is even, say a large power of 2, the values will be alternately odd and even. Then the spectral test in two dimensions should use $\partial = 2$, or equivalently, $m/2$ instead of m. Example 7 is a case where, in four dimensions, the maximal separation of **subgrids** is quadrupled by this modification, thus drastically reducing Knuth's figure of merit (next section) from 4.47 to **.073.** If we use the actual modulus instead of m/∂ in the **c(n)** formula, this figure reduces to **.018.** In practice, the user could destroy this "resonance" in the last two bits of his RNG simply by discarding every fifth value, assuming that his concern over **c(4)** arose from using quadruples of values. A different way to look at it is that Knuth's figure of merit may be too sensitive for large values, and that $\log \text{modulus}/e_{\min}$, the number of independent **bits, would** be better. Unfortunately, this would have to be multiplied by n for comparison between different dimensions.

Explanation of Example Printouts

The examples below indicate how the spectral test behaves in several interesting cases selected from the author's experiments. Each example is specified by a coded sequence of letters and numbers; the output, which was generated by slight variants of the MACSYMA programs in the appendix, can be understood as follows:

Procedure spec(a, m, n, nn) performs the spectral test on the LCRNG with multiplier a, modulus m, in dimensions n through nn,

In the leftmost column, an F indicates that the grid basis is used in the following transformation: an E indicates that the dual basis is used. The vector indicated by the second column is chosen as divisor, and all of the other vectors in that basis are "remaindered" by it, as described earlier. If all quotients are 0, no line is printed for that divisor and another divisor is tried. The vector printed out next consists of the c_j, indicating the size of the exhaustive search if no further reductions were possible. The integer following is the squared length of the shortest vector currently in the dual basis. Finally, TRUE indicates a new low exhaustive search volume, FALSE means no such luck. n consecutive divisor vectors without a TRUE means that the process is stuck. When both are stuck, the exhaustive search is performed, after which e_{min}^2, the square of the truly minimal dual vector length is printed (abbreviate this quantity d). On the next line are a, the multiplier: m, the modulus; n, the dimension; and finally, Knuth's figure of merit

$$c(n) = \frac{e_{min}^{n}\ \pi^{n/2}}{m\,(n/2)\,!}\ .$$

When both processes are stuck,' the exhaustive search rarely improves upon the current shortest vector,

Example 8

Here a particularly successful E process reduces c (the search vector) to one case: but for lack of a special check, it finds one more useless E step and an equally useless F step. Note **that when d** (4th column) fails to shrink, the only c vector entry (3rd column) which can change is the one indicated by the 2nd column, corresponding to the divisor vector,

```
(C8) spec(2654435789,2↑32,4,4)$
E 4 13967335, 5959262, 5959262, 15852981 35512803584646 TRUE
E 1 [851899, 5359262, 5959262, 15852981 35512803584646 TRUE
E 2 [116721, 120956, 816496, 2172061 666665718102 TRUE
E 3 [12611, 13068, 3066, 234681 7782610788 TRUE
E 4 [81, 84, 19, 153 325152 TRUE
E 1 [5, 84, 19, 151 328152 TRUE
E 2 [5, 7, 19, 151 3251.52 TRUE
E 3 [5, 7, 1, 15] 328152 TRUE
E 4 [1, 1, 0, 01 13558 TRUE
E 1 [0, 1, 0, 01 13558 TRUE
E 2 [0, 1, 0, 01 13558 FALSE (Next three divisors = 0 → stuck, try F)
F 3 [0, 1, 0, 01 13558 FALSE (It's no use, but F finds one nonzero quotient)
13558
2654435789 4294367296 4 0.211203495
TIRE= 40024 MSEC.
(40 sec. Well, nobody said MACSYMA was designed for number crunching)
```

Example 0, continued

Same problem using (slower) F process, which gets stuck after one step and is rescued by E. Again, some time is wasted in the end trying to reduce search to less than one case. Note that several elements of the c (search) vector can shrink in one step, even though d doesn't change.

```
(C00) SPEC(2654435789,2↑32,4,4)$
F 1[437552842, 541687612, 534854214, 5427929341 43196445834888O964 TRUE
E 2 [3967335, 1585298, 4849577, 49215581 35512803584646 TRUE
F 2 [839086, 1585238, 4849577, 43215581 35512803584646 TRUE
F   1 [839086, 588582, 4507021, 49215581 35512803584646 TRUE
F 2 1839086, 588582, 1502064, 39288751 35512803584646 TRUE
F  .3.[839086 588582  1502064, 38423081 35512803584646 TRUE
F 1 [839086,588582,938813, 33452561 35512803584646 TRUE
F  2 [517959, 363326, 534041, 20199701 13532031745394 TRUE
F  3 [417302  363326, 534041, 15822951 13532031745394 TRUE
F 1 [417302, 363326, 517959, 5555671 13532031745394 TRUE
F  3 [93262, 81199, 115757, 944011 675882938706 TRUE
F 4 [48720, 73941, 115757, 944011 675882938706 TRUE
F 1 [48720, 65388, 115757, 932621 675882938706 TRUE
F 2 [48386, 65537, 114965, 454433 666665718102 TRUE
F 4 [44078,64352, 74139, 454431 666665718102 TRUE
F 1 [31084, 45381, 43130, 32046] 331544488820 TRUE
F 2 [303, 443, 350, 3131 31640830 TRUE
F 3 [303, 421, 350,195] 31640830 TRUE
F 4 1133, 421, 313, 1353 31640830 TRUE
F 1 [133, 418, 193, 1401 31640830 TRUE
F 3 [133, 394, 193, 55] 31640830 TRUE
F 4 [127, 394, 55, 551 31640830 TRUE
F 3 [10, 35, 6, 6]  479910  TRUE
F 4 [10, 34, 6, 61 473910 TRUE
F 1 [10, 13, 6, 61 479910 TRUE
F 3 [10, 12, 6, 61 479310 TRUE
F 4 [10, 8, 6, 6]  479910  TRUE
F 2 [9, 8, 6, 61 479910 TRUE
F 3 [9, 7, 6, 6]479910  TRUE
F 4 [1, 0, 1, 1]13558  TRUE
F 1 El, 0, 1, 11 13558  FALSE
F 2 [0,0,0, 11 13558 TRUE
F 3 [0, B, 0, 13 13558 FALSE
F 1 [0, 0, 0, 13 13558  FALSE
F 2 [0, 0, 0, 11 13558 FALSE
E 3 [0, 0, 0, 1]  13558  FALSE
E 4 [0, 0, 0, 11 13558  FALSE
13558
2654435789 42945167296 4 0.211203495
TIME= 128120 MSEC.
```

Example 1

An example where the E process gets stuck with the search volume at 3307 cases, rescued by F, but only down to 314 cases, The first step achieving d = 1509 is remarkable for several reasons. 1509 will turn out to be the shortest squared length, leading to a spectacular c (6) of 8.27. Weirder is the fact that it managed to simultaneously reduce d and increase the search volume.
(Courtesy of David Hoaglin, Harvard Statistics Dept.)

```
(C1) spec(253634132,2^31-1,6,6)$
E 6 1104749768, 170403637, 170403637, 170403637, 170403637, 880781821
E   5   [25224117,                    23037399502827771                    TRUE
              41033803, 41033803, 41033803, 20465004, 212095401 1683772988642814
E   4   [10188858,                    16574916,                    T   R   U   E
E 3 [612718,996752, 16574916, 9462745, 8266495, 85672371 274727840407065 TRUE
E 2 [235476, 150306, 310236, 569053, 497115, 515200] 993514549513 TRUE
E 1 [14487, 28347, 21816248, 191048, 1979981 146738794278 TRUE
E 6 [7824,15303, 41224586, 36031, 373421 5219487568 TRUE
E 5 1148, 289, 22274143, 19458,   128521   1522250787   TRUE
E 4 [148,    28329, 42729155, 2545324      TRUE
E 3 [148, 289, 207, 110, 55, 24431      TRUE
E 2 [76, 116, 106, 10, 155, 24321      TRUE
E 1 [10,    31,    58, 73, 1243953      TRUE
E 6 [6,    19,    15,12, 1, 3319,10250 TRUE
                  13, 61 3882 TRUE
E 5 [6, 19, 17, 3, 3, 61 3882 TRUE
E 4 [6, 19, 17, 9, 3, 61 3882 FALSE
E 3 [4, 12, 1, 5, 2, 4] 1593 TRUE
E 2 [4,1, 1, 5, 2, 41 1593 TRUE
E 1 Cl, 1, 1, 5, 2, 41 1593 TRUE
E 6 [1, 1, 1, 5, 2, 11 1593 TRUE
E 5 [1, 1, 1, 5, 3, 11 1509 FALSE
E 4 [1, 1, 1, 3, 3, 11 1509 TRUE
E 2 [1, 2, 1, 3, 3, 11 1509 FALSE
E 6 [1, 2, 1, 3, 3, 11 1509 FALSE
F 3 [1, 1, 1, 1, 1, 13 1509 TRUE
E 2 [1, 1, 1, 1, 1, 11 1509 FALSE
F 2 [1, 1, 1, 1, 1, 1]1509 FALSE
F 1 [1, 1, 1, 1, 1, 11 1509 FALSE
F 6 [1, 1, 1, 1, 1, 1]1509 FALSE
F 5 [1, 1, 1, 1, 1, 11 1509   FALSE
1509
253634132 2147483647 6 X.2686827
TIME= 339425 MSEC.   (Long exhaust i ve search, )
```

Example 1, continued

Same problem, same process, but it somehow avoids getting stuck by cycling forwards through matrix of divisors instead of backwards. These qualities of large test value, many approach sequences, and large final search are all associated with highly isotropic grids.

```
E 0[104749768,SPEC[253634132,2^31-1,6,6]$
                  170483637, 170403637, 170403637, 170403637, 88078182]
E                      1       [487326,                          29037399502827771 TRUE
E 2[487326, 377125,  377125, 977125, 505056] 954773265689 TRUE
E 3 [160382, 377125,  977125, 977125,  505056] 954773265689 TRUE
E 4 [128769, 202225, 321579, 321579,  1662171  103413054855  TRUE
E 5[55490, 152870, 72252 , 242151,  1251621  58637107354  TRUE
E  6  [2790, 38273,569963,3197, 33076, 57509] 12379232699 TRUE
E 1 [63, 64,  1669,3519,1965,  7691  31315685  TRUE
E     2     163118, 568, 66, 25476     TRUE
E 3 [63,     18,18,  666,    25476     TRUE
E 4 [63,     218,56, 66,35476        TRUE
E 5 [24,     722, 128, 66, 25476     TRUE
E  6  [24,     74,  3,  31  5083 TRUE
E  1  [3,      4,4,  3,  51  5089 TRUE
E     2     13,  2, 51,  115, 1859 TRUE
E  3  [3,      52,  1, 13,  1859 FALSE
E     4     13,  2, 51,  113, 1859 TRUE
E  5  [3,      51,  1,  31  1859 TRUE
                1, 2, 11 1859 FALSE
E 1 [2, 5, 3, 1, 2, 11 1859  FALSE
E 2 [2,  1, 2, 1, 1, 11  1593  TRUE
E 3 [2, 1, 2, 1, 1, 13 1533 FALSE
E 6[ [1],   2, 3,1,  1,  21  1593 FALSE
                1, 1, 11  1593  TRUE
E  35 [1 [1,1,],1,  1,  1,11  1533 TRUE
E  2  [1,      1,1,  1,  11  1593 FALSE
                1, 1, 11 1509 FALSE
F 1[1,  1,  1,  1,  1,  11  1509  FALSE
1509
253634332  2147483647  6  8.2686827
TIME= 314353 MSEC.
```

Example 2

Somewhat unfortunate effects of using E process in incremental mode (on somewhat unfortunate multiplier distributed by IBM).
Here, we are using the leftover matrices, as in Refinement 3, but, contrary to suggestion, we are still using E instead of F.
Note the large search vectors created in the transition to next n.
About three times as many steps are required for n = 6 in incremental E mode than are required for doing n = 6 directly (not shown).
Had E gotten stuck earlier, F probably would have helped immensely.

For t-i = 2, the equivalence of the E and F processes will mean that the F process will rarely, if ever, find a nonzero quotient after the E process gets stuck.

```
(c2) spec(65533,2↑31,2,5)$
E 2 CI, 01 2147221544 TRUE
E 1 [1, 01 2147221544 FALSE
2147221544
65533 2147483643 2 3.1412093

E 1 [2, 278039,46338] 2147221544 TRUE
E 2 [0, 0, 101 118 TRUE
E 3 [0, 0, 1] 118 TRUE
E 2 [0, 0, 1] 118 FALSE
E 3 [0, 0, 11 118 FALSE
118
65533 2147483648 3 2.50024006E-6

E 1 [21, 879, 1465655, 101 118 TRUE
E 2 [21, 0, 1465655, 101 118 TRUE
E 3 [21, 0, 5, 103 118 TRUE
E 4 [21, 0, 5, 101 118 FALSE
E 1 [0, 0, 5, 103 116 TRUE
E 4 [0, 0, 5, 2] 116 TRUE
E 2 [0, 0, 5, 2] 116 FALSE
E 3 [0, 0, 1, 23 116 TRUE
E 4 [0, 0, 1, 11 116 TRUE
E 3 [0, 0, 1, 11 116 FALSE
E 4 [0, 0, 1, 11 116 FALSE
F 1 [0, 0, 1, 13 116 FALSE
F 2 [0, 0, 1, 11 116 FALSE
116
65533 2147483648 4 3.09211674E-5
```

```
E 2 [473, 2 310,6 , 3855060,  7842552,  103 116  TRUE
E  3  [473,      10,      7842552, 3854952, 101 116 TRUE
E  4 [473, 10,  3854352, 7842552, 116 TRUE
E  1  [10,  10,  3854952,       1927326, 101 116 TRUE
E  2  [10,          0,       1327326, 3854952, 103 116 TRUE
E  3  [10,          0,       1927326, 947330, 101 116 TRUE
E  4  [10,          0,       1927326, 947330, 103 116 TRUE
E  1  [0,           0,       473665, 947330, 101 116 TRUE
                              473665, 101 116 TRUE
E 3 [0, 0 [0, 2 3 2 0, 1 5232815, 473665, 101 116 TRUE
E  3  [0, 0,               116403, 57216, 101 116 TRUE
E  4  [0,          0,       116403, 57216, 101 116 TRUE
                              28603, 101 116 TRUE
E 3 [0,  0,  14053,  28603, 103 116 TRUE
E  4  [0, 0, 14059,  7030, 101 116 TRUE
E 3 [0,  0,  3450,  7030, 101 116 TRUE
E 4 [0,  0,  3450,  1720, 101 116 TRUE
E 3 [0,  0,  844,  1720, 103 116 TRUE
E  4 [0, 0, 844,  417, 101 116 TRUE
E 3 [0,  0,  208,  417, 103 116 TRUE
E 4 [0,0,  208,   105,  101 116 TRUE
E  5 [0,0, 208,   105,  101 116 FALSE
E 3 [0,  0,  47,  105,  101 116 TRUE
E 4 [0,  0,  47,  18, 101 116 TRUE
E 5 [0,  0,  47,  18, 101 116 FALSE
E 3 [0,  0,  4,  18, 103 116 TRUE
E 4 [0,  0,  4,  2,  101  116 TRUE
E 5 [0,  0,  4,  2,  11  116 TRUE
E 1 [0,  0,  4,  2,  11  116 FALSE
E 3 [0,  0,  1,  2,  11  116 TRUE
E 4 [0,  0,  1,  1,  11  116 TRUE
E 5 [0,  0,  1,  1,  1 1  116 FALSE
E 1  18, 0, 1, 1,   11  116 FALSE
E 2 [0,0,1,  1,   11  116 FALSE
E 3 [0,0,  1,  1,  11  116 FALSE
E 4 [0,0,1,  1,  13  116 FALSE
F 5 [0,0,  1,  1,  13  116 FALSE
F 1 [0,  0,  1,  1,  11 116 FALSE
F 3 [0,  0 ,1 ,1  11  116 FALSE
    [0, 0: 1, 1,  13  116  FALSE
116
65533 2147483648 5 3.552332E-4
TIME=256209 MSEC.
```

Example 2, continued

E Works best non-incrementally. For this particular multiplier of 65533, the cl of 116, which is such a disaster for n = 3, persists apparently through n = 9, for which it gets a very respectable c(9) of 3.00. (The author lacked the patience for the 1093 case final search,) For n = 10, d ≤ 64. This does not mean that it is a good idea to gobble 9-tuples from this generator and then discard a few in some pattern. To analyze the effects of this, perform the spectral test on the grid basis whose first vector skips the powers of a corresponding to the discarded values. More precisely, if you propose to discard the 3rd and 7th value out of every eight, delete the 3rd and 7th rows and columns from the eight dimensional basis matrices, Of course, if you are just discarding every other value, you could just run the regular

$$2$$

test on a instead of 3.

```
(C02) spec(65533,2^31,8,8)$
E 8[10647,65533,65533,65533,65533,65533,65533,65533,18707] 4294574090 TRUE
E 6[10647,65533,65533,65533,65533,65533,23328,18707] 4294574090 TRUE
E 5 [10647,65533,65533,65533,21402,23328,18707] 4294574090 TRUE
E 4 [4,27,65532,65536,25964,21402,23328,187071 4294574090 TRUE
10, 8, 9, 71 746 TRUE
E 3 14, 27, 5, 6, 10, 8, 9, 71 746 TRUE
E 2 [1,0, 2, 2, 4, 3, 3, 33 116 TRUE
E 1[0,0, 2, 2, 4, 3, 3, 31 116 TRUE
E 8 [0, 0, 2, 2, 4, 3, 3, 23 116 TRUE
E 7 [0,0, 2, 2, 4, 3, 3, 21 116 FALSE
E 6 [0, 0, 2, 2, 4, 3, 3, 2] 116 FALSE
E 5 [0, 0, 2, 2, 1, 3, 3, 2] 116 TRUE
E 4 [0, 0, 2, 1, 1, 3, 3, 2] 116 TRUE
E 3 [0. 0. 1. 1. 1 3 3. 2] 116 TRUE
E 2 10, 0, 1, 1, 13, 2],116 FALSE
E 8 [0,0,1, 1, 1, 3, 3, 1] 116 TRUE
E 7 [0,0, 1, 1, 1, 3, 1, 11 116 TRUE
E 6 [0, 0, 1, 1, 1, 1, 1, 11 116 TRUE
E 5 [0, 0, 1, 1, 1, 1, 1, 13 116 FALSE
E 8[0,0, 1, 1, 1, 1, 1, 13 116 FALSE
E 7 [0, 0, 1, 1, 1 , 1, 1, 1] 116 FALSE
E 6[0,0, 1, 1, 1, 1, 1, 11 116 FALSE
F 5 [0, 0, 1, 1, 1, 1, 1, 11 116 FALSE
F 4 [0,0, 1, 1, 1, 1, 1, 11 116 FALSE
F 3 [0, 0, 1, 1, 1, 1, 1, 11 116 FALSE
F 2 [0,0, 1, 1, 1, 1, 1, 1] 116 FALSE
F 1 [0, 0, 1, 1, 1, 1, 1, 11 116 FALSE
F 8 [0,0, 1, 1, 1, 1, 1, 11 116 FALSE
116
65533 2147483648 8 0.34220817
TIME= 484930 MSEC,
```

Example 3

The F process in incremental mode, same a = 65533. Note nice, small transitions to next n.

```
(C3) spec (65533,2^31,2,8)$

F 1 [1,0] 2147221544 TRUE
F 2 [1, 0] 2147221544 FALSE
2147221544
65533 2147483648 2 3.1412093

F 3 [0,0, 103 118 TRUE
F 1 [0, 0, 41 118 TRUE
F 2 [0, 0, 11 118 TRUE
F 1 [0, 0, 11 118 FALSE
F 2 [0,0, 11 118 FALSE
E 3 [0, 0, 11 118 FALSE
118
65533 2147483648 3 2.50024006E-6

F 4 [0, 0, 5, 10] 116 TRUE
F 1 [0, 0, 1, 1] 116 TRUE
F 2 [0, 0, 1, 1] 116 FALSE
F 3 [0, 0, 1, 1] 116 FALSE
F 1 [0, 0, 1, 1] 116 FALSE
E 3 [0, 0, 1, 1] 116 FALSE
E 4 [0, 0, 1, 1] 116 FALSE
116
65533 2147483648 4 3.09211674E-5

F 5 [0,0, 4, 4, 101 116 TRUE
F 1 [0,0,1, 1, 21 116 TRUE
F 2 [0, 0, 1, 1, 2] 116 FALSE
F 3 [0, 0, 1, 1, 11 116 TRUE
F 4 [0,0,1, 1, 11 116 FALSE
F 1 [0,0, 1, 1, 11 116 FALSE
F 2 [0, 0, 1, 1, 11 116 FALSE
E 4 [0,0, 1, 1, 1] 116 FALSE
E 5 [0, 0, 1, 1, 11 116 FALSE
E 3 [0, 0, 1, 1, 11 116 FALSE
,116
65533 2147483648 5 3.552332E-4

F 6 [0, 0, 2, 5, 1, 10] 116 TRUE
F 1 [0, 0, 1, 2, 1, 4] 116 TRUC
F 2 [0, 0, 1, 1, 1, 1] 116 TRUE
F 3 [0, 0, 1, 1, 1, 1] 116 FALSE
F 6 [0, 0, 1, 1, 1, 1] 116 FALSE
E 3 [0, 0, 1, 1, 1, 1] 116 FALSE
116
65533 2147483648 6 3.75614646E-3
```

(6 is shorter than 4 or 5! 155 secs so far)

```
F 1 [0, 0, 0, 4, 3, 1, 3, 10] 116 TRUE
F 2 [0, 0, 1, 1, 1, 1, 11 116 TRUE
F 3 [0, 0, 1, 1, 1, 1, 11 116 FALSE
F 4 [0, 0, 1, 1, 1, 1, 11 116 FALSE
F 1 [0, 0, 1, 1, 1, 1, 1] 116 FALSE
                1, 1, 1, 11 116
E 3 [0, 0, 1, 1, 1, 1, 13 116 FALSE
116             1, 1, 3, 1] 116 FALSE

65533 2147483648 7 0.036987356

F 1 [0, 0, 0, 5, 5, 1, 3,3, 10] 116 TRUE
F 2 [0, 0, 1, 1, 1, 1, 21 116 TRUE
F 3 [0, 0, 1, 1, 1, 1, 21 116 FALSE
F 4 [0, 0, 1, 1, 1, 1, 1, 11 116 TRUE
F 6 [0, 0, 1, 1, 1, 1, 1, 11 116 FALSE
F 7 [0, 0, 1, 1, 1, 1, 1, 11 116 FALSE
F 1 [0, 0, 1, 1, 1, 1,1, 1] 116 FALSE
F 2 [0, 0, 1, 1, 1, 1,1, 11 116 FALSE
E 5 [0, 0, 1, 1, 1, 1, 1, 11 116 FALSE
E 6 [0, 0, 1, 1, 1, 1, 1] 116 FALSE
E 7 [0, 0, 1, 1, 1, 1,1, 1] 116 FALSE
E 8 [0, 0, 1, 1, 1, 1, 1] 116 FALSE
116             1, 1, 1, 1, 3] 16 FALSE

65533 2147483648 8 0.34220817
(Approximate time: 700 sec.)
```

Example 4

A particularly good multiplicator from an extensive search by F. Janssens. Note helpful E on n = 4.

```
(C4) spec (1690613,2^32,2,5)$
F 1 [318,152]656553647776 7 R U E
F 2 C 1 , 13] 4934360282 TRUE
F 1 [1,1] 4934360252 TRUE
4934360282
1690613 4294967296 2 3.60928246

F 3 [12382, 343 , 29323] 860481298 TRUE
F 1 [5011, 139, 1848] 140955560 TRUE
F 2 [6. 96, 2.8 1 67354506 TRUE
F 3 [3, 6, 16]22357546 TRUE
F 1 [2, 3, 6] 7352462 TRUE
F 2 [1, 2, 0 1 2162558 TRUE
F 3 [0, 1, 0 1 2162558 TRUE
F 1 [0, 1, 0] 2162558 FALSE
E 2 [0, 1, 0]2162558 FALSE
2162558
1690613 4294367296 3 3.1015612

F 4 1333, 173, 19, 8051 649062 TRUE
F 1 [329, 158, 19, 1351 631750 TRUE
F 2 [4, 58, 6, 81 84994 TRUE
F 3 [2, 2, 6, 1] 84572 TRUE
F 4 [1, 1, 2, 13 84572 TRUE
F 1 [1, 1, 1, 1] 84572 TRUE
E 2 [0, 1, 1, 11 62910 TRUE
E 3 [0, 1, 1, 11 62910 FALSE

1690613 4294967296 4 4.547254

F 5 [79, 26, 83, 39, 183] 33490 TRUE
F 1 [65, 22, 3, 32, 131 22994 TRUE
F 2 [2, 22, 3, 10 , 2] 22994 TRUE
F 3 [1, 5, 2, 3, 11 15110 TRUE
F 4 [1, 1, 1, 2, 11 8834 TRUE
F 5 [1, 1, 1, 2, 1 1 8834 FALSE
F 1 [1, 1, 1, 1, 0]6226 TRUE
F 2 [0, 1, 0, 0, 01 4.1'24 TRUE
F 3 [0, 1, 0, 0, 0 1 4224 FALSE
F 4 [0, 1, 0, 0, 03 4224 FALSE
F 5 [0, 1, 0, 0, 01 4224 FALSE
E 3. [0, 1, 0, 0, 0] 4224 FALSE
E 4 · [0, 1, 0, 0, 0]4224 FALSE
E 2 [0, 1, 0, 0, 03 4224 FALSE
4224
1690613 4294967296 5 1.421178
TIME= 121136 MSEC.
```

Example 5

Incremental F mode on an old standby.

```
(C5) spec(5↑15,2↑35,2,8)$
F 1 [24567628, 1404835] 48269768083184842 TRUE
F 2 [334427, 685430] 11490836629925504 TRUE
F 1 [16576, 821] 28229871184010 TRUE
F 2 [25, 144] 868180541344 TRUE
F 1 [7, 2] 78377938346 TRUE
F 2 [0, 1] 22078865098 TRUE
F 1 [0, 1] 22078865098 FALSE
22078865098
30517578125 34359738368 2 2.0187232

F 3 [34843, 41197, 99637] 992762
F 1 [12957, 2362, 1819] 13729732                0238 TRUE
F 2 [1142, 2362, 543] 1372973330                30 TRUE
F 3 [15, 51, 147] 100694474 TRUE                TRUE
F 1 [15, 5, 9] 100694474 TRUE
F 2 [2, 2, 3] 14523936 TRUE
F 3 [1, 2, 3] 14523936 TRUE
F 1 [1, 1, 0] 10274746 TRUE
F 3 [1, 1, 0] 10274746 FALSE
E 1 [1, 1, 0] 10274746 FALSE
10274746
30517578125 34359738368 3 4.0150921

F 4 [14 , 52, 407, 1179] 1390614 TRUE
F 1 [14 , 6, 12, 31] 1390614 TRUE
F 2 [1,  3, 4, 15] 333602 TRUE
F 3 [1,  2, 3, 1] 169714 TRUE
F 4 [0,  1, 3, 1] 169714 TRUE
F 1 [0,  0, 2, 1] 169714 TRUE
F 2 [0,  0, 1, 1] 169714 TRUE
F 4 [0,  0, 1, 1] 167558 FALSE
F 2 [0,  0, 1, 1] 167558 FALSE
167558
30517578125 34353738365: 4 4.0322757

E 1 [104, 22, 34, 138,287182664 TRUE
     1104, 22, 34, 33, 271 82664 TRUE
F 3 [1, 05, 7, 8, 31 42752 TRUE
F 4 [1, 0 , 2, 0, 13 5344 TRUE
F 5 [0, 0 , 2, 0, 1] 5844 FALSE
F 1 [0,    1, 0, 13 5344 TRUE
F 2 [0, 00, 1, 0, 13 5844 FALSE
F 4 [0, 0 , 1, 0, 03 5844 TRUE
F 1 [0,    1, 0, 01 5844 FALSE
5844    0, 1, 0, 03 5844 FALSE

30517578125 34359738368 5 0.39996696
```

```
F 6 [25, 13, 20, 33, 30, 76] 5844 TRUE
F 1 [25, 11, 5, 8, 5, 1] 5844 TRUE
F 2 [2, 11, 5, 3, 5, 1] 5844 TRUE
F 3 [1, 2, 4, 1, 1, 1] 3890 TRUE
F 4 [1, 1, 2, 1, 1, 1] 2890 TRUE
F 5 [0, 1, 2, 1, 1, 1] 2890 TRUE
F 6 [0, 1, 2, 1, 1, 1] 2890 FALSE
F 1 [0, 1, 2, 1, 0, 1] 2890 TRUE
F 2 [0, 1, 0, 1, 0, 0] 2592 TRUE
F 3 [0, 1, 0, 1, 0, 0] 2592 FALSE
F 5 [0, 1, 0, 1, 0, 0] 2592 FALSE
E 4 [0, 1, 0, 1, 0, 0] 2592 FALSE
2592
30517578125 34359738368 6 2.6191089

F 7 [9, 3, 7, 11, 1, 0, 29] 886 TRUE
F 1 [6, 2, 1, 2, 0, 0, 2] 508 TRUE
F 2 [1, 2, 1, 0, 0, 0, 1] 508 TRUE
F 3 [1, 1, 1, 0, 0, 0, 1] 508 TRUE
F 4 [1, 1, 1, 0, 0, 0, 1] 508 FALSE
F 5 [1, 1, 0, 0, 0, 0, 1] 508 TRUE
F 6 [1, 1, 0, 0, 0, 0, 1] 508 FALSE
F 7 [1, 1, 0, 0, 0, 0, 1] 508 FALSE
F 2 [1, 1, 0, 0, 0, 0, 1] 508 FALSE
F 3 [1, 1, 0, 0, 0, 0, 0] 508 TRUE
F 4 [1, 1, 0, 0, 0, 0, 0] 508 FALSE
F 5 [1, 1, 0, 0, 0, 0, 0] 508 FALSE
F 6 [1, 0, 0, 0, 0, 0, 0] 508 TRUE
F 7 [1, 0, 0, 0, 0, 0, 0] 508 FALSE
F 4 [1, 0, 0, 0, 0, 0, 0] 508 FALSE
F 5 [1, 0, 0, 0, 0, 0, 0] 508 FALSE
E 7 [1, 0, 0, 0, 0, 0, 0] 508 FALSE
E 2 [1, 0, 0, 0, 0, 0, 0] 508 FALSE
508
30517578125 34359738368 7 0.406306185

F 8 [3, 9, 10, 2, 7, 7, 1, 22] 508 TRUE
F 1 [3, 2, 3, 1, 2, 2, 1, 6] 508 TRUE
F 2 [3, 2, 1, 1, 2, 1, 1, 1] 508 TRUE
F 3 [3, 2, 1, 1, 2, 1, 1, 1] 508 FALSE
F 4 [2, 2, 1, 1, 1, 1, 1, 1] 508 TRUE
F 5 [1, 2, 1, 1, 1, 1, 1, 1] 508 TRUE
F 6 [1, 0, 1, 1, 1, 1, 1, 1] 508 TRUE
F 7 [1, 0, 1, 1, 1, 1, 1, 1] 508 FALSE
F 8 [1, 0, 1, 1, 1, 1, 1, 1] 478 FALSE
F 1 [1, 0, 1, 1, 1, 1, 1, 1] 478 FALSE
F 2 [1, 0, 1, 1, 1, 1, 1, 1] 478 FALSE
F 4 [1, 0, 1, 1, 1, 1, 1, 1] 478 FALSE
F 5 [1, 0, 1, 1, 1, 1, 1, 1] 478 FALSE
F 6 [1, 0, 1, 1, 1, 1, 1, 1] 478 FALSE
E 7 [0, 0, 1, 1, 1, 1, 1, 1] 414 TRUE
F 3 [0, 0, 1, 1, 1, 1, 1, 1] 414 FALSE
F 5 [0, 0, 1, 1, 1, 1, 1, 1] 414 FALSE
E 8 [0, 0, 1, 1, 1, 1, 1, 1] 414 FALSE
E 7 [0, 0, 1, 1, 1, 1, 1, 1] 414 FALSE
414
30517578125 34359738368 8 3.4700827
TIME= 745703 MSEC.
```

Example 6

A variation of the program which doesn't bother to compute search volume vector except when E or F process gets stuck, where stuck is defined to be all quotients 0, rather than n consecutive failures to reach a new low search volume. From this data it appears that 65533 is the worst multiplier in a bad neighborhood.

```
(C6) for a:65533-16 step 8 thru 65533+16 do spec(a,2^31,2,5)$
F [1, 0] 2135723368 TRUE
E [1, 0] 2135723368 FALSE
2135723368
65517 2147483648 2 3.12438837

YOU HAVE RUN OUT OF LIST SPACE.  DO YOU WANT MORE?
TYPE ALL; NONE; OK; A LEVEL-NO. OR THE NAME OF A SPACE.
list;
^A 9383 MSEC.
(MACSYMA-BREAK)
←dynamalloc:not(dynamalloc);
TRUE
+-exit:
EXITED FROM THE BREAK        (Ahh, interactive languages)
F [0, 0, 1] 131766 TRUE
E [0, 0, 1] 131766 FALSE
131766
65517 2147483648 3 0.003296174

F [0, 1, 0, 0] 34622 TRUE
E [0, 1, 0, 0] 34622 FALSE
34622
65517 2147483648 4 2.75450945

F [1, 1, 0, 0, 0] 3106 TRUE
E [1, 1, 0, 0, 0] 3106 FALSE
3106
65517 2147483648 5 1.31786923

F [1, 0] 2143555880 TRUE
E [1, 0] 2143555880 FALSE
2143555880
65525 2147483648 2 3.1358467

F [0, 0, 1] 15126 TRUE
E [0, 0, 1] 15126 FALSE
15126
65525 2147483648 3 3.6286475E-3

F [0, 0, 1, 1] 15126 TRUE
E [0, 0, 1, 1] 15126 FALSE
15126
65525 2147483648 4 0.5257607

F [0, 0, 1, 0, 0] 2796 TRUE
E [0, 0, 1, 0, 0] 2796 FALSE
2796
65525 2147483648 5 1.01323817
```

```
F [1, 0: 2147221544 TRUE
E [1, 0] 2147221544 FALSF
2147221544
65533 2147483648 2 3.1412093

F [0, 0, 11 118 TRUE
E [0, 0, 1] 118 FALSE
118
65533 2147483648 3 2.50024006E-6

F [0, 0, 1, 1] 116 TRUE
E [0, 0, 1, 1] 116 FALSE
116
65533 2147483648 4 3.09211674E-5

E [0, 0, 1, 1, 13 116 TRUE
116 1, 11 116 FALSE

65533 2147483648 5 3.552332E-4

F [1, 01 2146697320 TRUE
E [1, 03 2146697320 FALSE
2146697320
65541 2147483648 2 3.1404424

F [0, 0, 1] 726 TRUE
E [0, 0, 1] 726 FALSE
726
65541 2147483648 3 3.81560716E-5

F [0, 0, 1, 11 726 TRUE
E [0, 0, 1, 1] 726 FALSE
726
65541 2147483648 4 1.2111924E-3

F [0, 0, 1, 1, 1] 726 TRUE
E [0, 0, 1, 1, 1] 726 FALSE

65541 2147483648 5 0.034810489

F [1, 03 2141984744
E [1, 03 2141984744 TRUE
2141984744 FALSE
65549 2147483648 2 3.13354826

F [0, 0, 1] 29238 TRUE
E [0, 0, 11 29238 FALSE
29238
65549 2147483648 3 9.75169825E-3

F [0, 0, 1, 11 29238 TRUE
E [0, 0, 1, 13 29238 FALSE

65549 2147483648 4 1.96442376

E [1, 1, 1, 1, 01 5132 TRUE
5132 1, 01 5132 FALSE

65549 2147483648 5 4.6247172
MSEC.
```

Example 7

(Program modified as in Example 6). Another good Janssens number but look what happens if we use m/4 instead of m for n = 4 in 2nd set', so as to consider only independent tuples. Taking every 4th point quadrupled the length of the grid vector which was already connecting the most widely separated subgrids--the worst possible outcome. In the dozen or so other cases tried, the degradation uas much less severe.

```
(C7) spec(1664525,2↑32,2,4)$
F [1, 13 4938916874 TRUE
E [1, 11 4938916874 FALSE
4338916874
1664525  4294967296  2  3.61261544

F E1, 0, 01 2322494 TRUE
E [1, 0, 01 2322434 FALSE
2322494
1664525  4234367236  3  3.4519195

F [0, 1, 1, 01 63712 TRUE
E [0, 1, 1, 03 63712 FALSE
63712
1664525  4234967236  4  4.6639335
TIME= 52365 MSEC.

(C07) spec(1664525,2↑30,2,4)$
F [1, 03 310518218 TRUE
E C1, 01 310518218 FALSE     (Th s data really applies to mod 2^31.)
310518218
1664525  1073741824  2  0.90852543

F [0, 1, 0] 412832 TRUE
E [0, 1, 01 412832 FALSE     (Th s data really applies to mod 2^30.)
412832
1664525  1073741824  3  1.0347817

F [0, 0, 0, 11 3382 TRUE
E [0, 0, 0, 13 3982 FALSE    (This data really applies to mod 2^32.)
3982
1664525  1073741824  4  0.072873961
TIME= 44372 MSEC.

(C37) 3982*16;
TIME= 6 MSEC.
(D37)                        63712
```

Example 8

Some pathology of incremental F mode, For a = 663608301, n = 7, E tried tobeat d = 390 and tripled the exhaustive search instead, On the next case, a = 663608303, n = 7, E manages to win. Later, when a = 663608933, n = 7, we have a serious failure of E to find the smallest d. Instead, for the first time in the author's experience, a 2 appears in the final search vector, and the (very) exhaustive search actually finds a smaller d. In the final case, a = 663608941, (a multiplicator constructed by Dieter to have a good two dimensional serial test), n = 7, we have another minor misstep ofE. Note unlucky 7. Actually, these problems are just hard to find for small n. The underlying cause of this pathology is that, were it not for the inefficiency, we would be better off hunting a small d using E first, then trying to reduce the search with F.It is an empirical fact that Fincreases d far more rarely (if at all) than E increases the search volume. When the author tried to run Eand F alternately until both were stuck, they fell into a period 2 oscillation modifying one vector in the case a = 663608333, n = 6, In an effort to see what would happen anyway on the case n = 7, the author manually intervened to stop the n = 6 loop. Somewhat unfortunately, due to the additional reduction of the n = 6 bases, this "ping-pong" variation was never confronted with the matrix which led E to create the [1, 1, 1, 2, 1, -1, 13, but instead F came right up with five 1s and two 0s and d = 442.

```
(C8) FOR A: 663608901 STEP 8 DO SPEC(A,2↑32,2,7);
F [0, 13 3397158986 TRUE
E [0, 1] 3397158986 FALSE
3397158986
663608901  4234367236  2  2.4848827

F [1, 0,  01 1490074 TRUE
E El, 0,  03 1490074 FALSE
1490074
663608901  4294967296  3 1.77394624

F [0, 0,  1,  01 13408 TRUE
E [0, 0,  1,  01 13408 FALSE
13408
663608901  4294967296  4  0.20655603

F [0, 1,  0,  0,  03 1078 TRUE
E [0, 1,  0,  0,  03 1078 FALSE
1078
663608901  4294967296  5  0.046761183

F [0, 1,  1, 0,  0,  01 1078 TRUE
E [0, 1,  1, 0,  0,  01 1078 FALSE
1078
663608901  4294967296  6  1.50728331

F Cl, 0,  1, 0,  0,  0,  01 330 TRUE
E [1, 1,  1, 0,  0,  0,  01 390 FALSE
390
663608901  4234967296  7  1.28868447
```

```
F [0, 11  626928730  TRUE
E [0, 11  626928730  FALSE
626928730
663608909  4294967296  2  0.45857269

F [1, 0,  03  465630  TRUE
E [1, 0,  01  465630  FALSE
465630
663608909  4294967236  3  00303877604

F [1, 1,  1,  03  53172  TRUE
E El,  1,  1,  01  53172  FALSE
53172
663608909  4234967296  4  3.2484476

F [0, 0, 0,  1,  11  6314  TRUE
E [0, 0,  0,  1,  11  6314  FALSE
6314
663608909  4294967296  5  3.8823978

E [1, 0,  1,  1,  0,  03  1406  TRUE
1406          1,  0,  01  1406  FALSE

663608909  4294967296  6  3.34421745

E [1, 1,  1,  0,  1,  1,  11  996  TRUE
F [1,  1,  1,  0,  1,  0,  11  534  TRUE
E [1,  1,  1,  0,      0,  11  534  FALSE
--.                1,  0,  11  534  FALSE
534
663608909 4294967296 7 3.8709251

          . .

F [1,  03  3236682938  TRUE
E Cl,  0] 3236682938  FALSE
3236682938
663608933  4294967236  2  2.367501

F [0, 0,  13  311352  TRUE
E [0, 0,  13  311352  FALSE
311352
663608933  4294967296  3  0.16943623

F [0, 1,  0,  0]  13562  TRUE
E [0, 1,  0,  01  13562  FALSE
13562
663608933  4294367296  4  0.21132812

F [1,  1,  1,  1,  01  7298  TRUE
E [1,  1,  1,  1,  03  7298  FALSE
7298
663608933  4294967296  5  5.5763277
```

E [1, 0, 0, 0, 0, 1] 958 TRUE
958 0, 0, 11 958 FALSE

663608333 4294367296 6 1.0578767

F Cl, 1, 1, 1, 1, 13
E Cl, 1, 1, 2, 1,1 1,, 11 746 TRUE

663608933 4294967296 7 1.99709308

F [1, 03 1606825210 TRUE
E Cl, 03 1606825210 FALSE
1606325210
663608941 4294967296 2 1.17532684

F [1, 0, 01 477530 TRUE
E [1, 0, 01 477530 FALSE
477530
663608941 4294367236 3 0.321832295

F Cl, 0, 0, 01 45014 TRUE
E Cl, 0, 0, 03 45014 FALSE

663608941 4294967296 4 2.3281182

F [0, 0, 0, 0, 11 2582 TRUE
E [0, 0, 0, 0, 1] 2582 FALSE

663608341 4294367296 5 0.41517307

E [1, 0, 1, 0, 0, 01 788 TRUE
788 0, 0, 01 788 FALSE

663608341 4234367236 6 0.58873131

E [0, 0, 0, 0, 0, 1, 0] 338 TRUE
338 0, 1, 11 338 FALSE

663608941 4234367236 7 0.78096013

. . .

Program Listing

```
time: translate:modedeclare([value(better),value(tr)],boolean,
Ea, ap,modulus,n,i,j,k,gg,d,mm,nn,p], integer)$
radprodexpand: false$

spec(a,modulus,n,nn):=(d:mm: (a*modulus)↑(2*n), scalarmatrixp:false,
f:modulus*e: ident(n-1), ap:1,
for k thru n-1 do(e[k,1]:-f[1,k]:ap,ap:ratdisrep(rat(a*ap))),
e[1,1]:modulus,
for n:n thru nn do block(mm:mm*(a*modulus)↑2,
 e: transpose(addrow(transpose(addrow(e,ematrix(1,1:n-1,-ap,1,1))),
                     ematrix(1,n,1,1,n))),
 f:addrow(transpose(addrow(f:transpose(f),ap*f[1])),
          ematrix(1,n,modulus,1,n)),
 scalarmatrixp:ap:ratdisrep(rat(a*ap)),
 c:reverse(z:reverse(q:ematrix(1,n,0,1,1)[1])),
 t, if tr('f,'e) or tr('e,'f) then go(t) else ktn,
 u, if (z[k]:z[k]+1)>c[k] then go(w),
 v, if (k:k+1)>n then d:min(d,(b:z.e).transpose(b)) else

 (z[k]:-c[k],go(v)),
 w, if (k:k-1)>0 then go(u) else
  print(a,modulus,n,ev((print(d)*%pi)↑(n/2)/modulus/gamma(n/2+1),numer)))$

tr(ff,ee):=for j do (i:1+remainder(i,n),
gg: (g:(vf:ev(ff))[i]).b:transpose(g),
for k thru n do q[k]:entier(vf[k].b/gg+1/2),
q[i]:0, if q#z and better() then (mm:p,return(true))
else if j=n then return(false))$

better():=([],ee::substinpart((q.ve)[1]+ve[i],ve:ev(ee),i),
ff::vf-transpose(q).g,
for k thru n do d:min(d,e[k].transpose(e[k])),
p:1, for k thru n do (c[k]:isqrt(entier(d*f[k].transpose(f[k])/modulus↑2)),
                      p:(2*c[k]+1)*p),
print(ff,i,c,d,is(p<mm)))$
```

Notes

Thi s was done over the ARPANET on the MIT MACSYMA System, to which I am indebted both for the language and the machine time,

The first three lines are optional. They turn on the timer and make type declarations for efficiency.

d and mm are best e_{min}^{2} and search volume so far.

scalarmatrixp: unfortunately, MACSYMA can regard a 1 by 1 matrix as a scalar, but not vice versa.

f: the grid basis matrix, e: the dual, ap: consecutive powers of a mod m. modulus is a special variable such that ratdisrep(rat(x)) is the residue of least magnitude mod modulus.

transpose (addrow (transpose(addrow...: MACSYMA lacks an addcolumn. reverse is a fast way to copy a list of 0s. gamma (n/2+1): pedantry for (n/2)!. %pi: π. substinpart(e,r,i):replaces ith row of e with r, evaluating r first,

The Modified Program

```
time: translate:modedeclare([value(better),value(tr)],boolean,
[a,ap,modulus,n,i,j,k,gg,d,mm,nn,p],integer)$
radprodexpand:false$

spec(a,modulus,n,nn):=(d:mm:(a*modulus)↑(2*n), scalarmatrixpr false,
f:modulus*e:ident(n-1), ap:1,
for k thru n-1 do(e[k,1]:-f[1,k]:ap, ap:ratdisrep(rat(a*ap))),
e[1,1]:modulus,
for n:n thru nn do block(mm:mm*(a*modulus)↑2,
 e: transpose(addrow(transpose(addrow(e,ematrix(1,i:n-1,-ap,1,1))),
                     ematrix(1,n,1,1,n))),
 f: addrow(transpose(addrow(f: transpose(f),ap*f[1])),
           ematrix(1,n,modulus,1,n)),
 scalarmatrixp:ap:ratdisrep(rat(a*ap)),
 c:reverse(z:reverse(q:ematrix(1,n,0,1,1)[1])),
 t, tr('f,'e), if tr('e,'f) then go(t) else k:n,
 u, if (z[k]:z[k]+1)>c[k] then go(u),
 v, if (k:k+1)>n then d:min(d,(b:z.e).transpose(b)) else (z[k]:-c[k],go(v)),
 w, if (k:k-1)>0 then go(u) else
  print(a,modulus,n,ev((print(d)*%pi)↑(n/2)/modulus/gamma(n/2+1),numer))))$

tr(ff,ee):=(for j thru n do (i:1+remainder(i,n),
 gg: (g:(vf:ev(ff))[i]).b:transpose(g),
 for k thru n do q[k]:entier(vf[k].b/gg+1/2),
 q[i]:0,if q#z then (j:1, ee:: substinpart((q.ve)[1]+ve[i],ve:ev(ee),i),
                     ff::vf-transpose(q).g)),
if better() then (mm:p, true))$

better():=(for k thru n do d:min(d,e[k].transpose(e[k])),
p:1, for k thru n do (c[k]:isqrt(entier(d*f[k].transpose(f[k])/modulus↑2)),
                      p: (2*c[k]+1)*p),
print(ff,c,d,is(p<mm)))$
```

Note

Since both of these versions use the F process incrementally, it is usual ly best to start with n = 2. To convert either version to the E process, just change the line beginning

```
tr(ff,ee):= ... to read tr (ee, ff) ...
```