The Problem with the Linpack Benchmark Matrix Generator

June 28, 2008

Jack Dongarra
Department of Electrical Engineering and Computer Science, University of Tennessee
Oak Ridge National Laboboatory
University of Manchester
Julien Langou
Department of Mathematical and Statistical Sciences, University of Colorado Denver

Abstract:
We characterize the matrix sizes for which the Linpack Benchmark matrix generator constructs a matrix with identical columns.

1 Introduction

Since 1993, twice a year, a list of the sites operating the 500 most powerful computer systems is released by the TOP500 project [3]. A single number is used to rank computer systems based on the results obtained on the *High Performance Linpack (HPL) Benchmark*.

The High Performance Computing Linpack Benchmark consists of solving a dense linear system in double precision, 64–bit floating point arithmetic, using Gaussian elimination with partial pivoting. The ground rules for running the benchmark state that the supplied matrix generator, which uses a pseudo–random number generator, must be used in running the HPL benchmark. The supplied matrix generator can be found in HPL [2] which is an implementation of the High Performance Computing Linpack Benchmark. In the HPL benchmark program, the correctness of the computed solution is established and the performance is reported in floating point operations per sec (Flops/sec). It is this number that is used to rank computer systems across the world in the TOP500 list.

In May 2007, a large high performance computer manufacturer ran a twenty-hour-long High Performance Linpack benchmark. The run fails with the output result:

```
|| A x - b ||_oo / ( eps * ||A||_1  * N ) = 9.22e+94 ...... FAILED
```

It turned out that the manufacturer chose n to be $n = 2,220,032 = 2^{13} \cdot 217$. This was a bad choice. In this case, the Linpack Benchmark matrix generator produced a matrix A with identical columns. Therefore the matrix used in the test was singular and one of the checks of correctness determined that there was a problem with the solution and the results should be considered questionable. The reason for the suspicious results was neither a hardware failure nor a software failure but a predictable numerical issue.

In this manuscript, we explain why the Linpack Benchmark matrix generator can generate matrices with at least two identical columns for particular matrix sizes n. We name this set of integers $\mathcal{S}$. We characterize and give a simple algorithm to determine if a given n is in $\mathcal{S}$.

Definition 1 *We define $\mathcal{S}$ as the set of all integers such that the Linpack Benchmark matrix generator produces a matrix with at least two identical columns.*

In Table 1, we give the 40 smallest integers in $\mathcal{S}$.

Some remarks are in order.

65,536	98,304	131,072	147,456	163,840
180,224	196,608	212,992	229,376	245,760
262,144	270,336	278,528	286,720	294,912
303,104	311,296	319,488	327,680	335,872
344,064	352,256	360,448	368,640	376,832
385,024	393,216	401,408	409,600	417,792
425,984	434,176	442,368	450,560	458,752
466,944	475,136	483,328	491,520	499,712

Table 1: The 40 matrix sizes smaller than $500,000$ for which the Linpack Benchmark matrix generator will produce a matrix with identical columns.

1. If n is in $\mathcal{S}$, then the matrix generated by the Linpack Benchmark generator has at least two identical columns. If n is not in $\mathcal{S}$, the matrix has no identical columns; however we do not claim that the matrix is well-conditioned, we do not even even claim that the matrix is nonsingular. Therefore not being in $\mathcal{S}$ is not a sufficient condition for being safe of numerical failures.

2. HPL checks whether a zero-pivot occurs during the factorization and reports it to the user. However due to rounding errors, even if the initial matrix has two identical columns, exact-zero pivots hardly ever occur in practice. Consequently, it is difficult for benchmarkers to distinguish between numerical failures and hardware/software failures.

3. The matrix size for the #1 entry in the TOP500 list of June 2008 was $2,236,927$ which is between 2^{21} and 2^{22}. The smallest matrix size in the TOP 500 list of June 2008 was $273,919$ which is between 2^{18} and 2^{19}.

4. To verify the result, the input matrix and right-hand side are regenerated. The following scaled residuals are computed (ε is the relative machine precision):

$$r_n = \frac{||Ax-b||_\infty}{n\varepsilon||A||_1} \tag{1}$$

$$r_1 = \frac{||Ax-b||_\infty}{\varepsilon||A||_1||x||_1} \tag{2}$$

$$r_\infty = \frac{||Ax-b||_\infty}{\varepsilon||A||_\infty||x||_\infty} \tag{3}$$

   A solution is considered numerically correct when all of these quantities are less than a threshold value of 16.

   The last quantity (r_∞) corresponds to a backward error in the infinite norm. The last two quantities (r_∞, r_1) are independent of the condition number of the coefficient matrix A and should always be less than a threshold value of the order of 1 (no matter how ill-conditioned A is). The first quantity (r_n) is proportional to the inverse of the condition number of the coefficient matrix A so this quantity can be arbitrarily large if the coefficient matrix is not well-conditioned.

2 How the Linpack Benchmark matrix generator constructs a pseudo–random matrix

The pseudo–random coefficient matrix A from the Linpack Benchmark matrix generator is generated by the HPL subroutine `HPL_pdmatgen.c`. In this subroutine, the pseudo–random generator uses a linear congruential algorithm [1, §3.2]

$$X(n+1) = (a * X(n) + c) \bmod m,$$

with $m = 2^{31}$. From [1, §3.2], we know that the maximum period of the sequence is at most m, and in our case, with HPL's parameters a and c, we can check that we indeed obtain the maximal period 2^{31}. This provides us with a periodic sequence s such that $s(i+2^{31}) = s(i)$, for any $i \in \mathbb{N}$. HPL fills its matrices with pseudo–random numbers by columns using this sequence s starting with $A(1,1) = s(1)$, $A(2,1) = s(2)$, $A(3,1) = s(3)$, and so on.

Definition 2 *We define a Linpack Benchmark matrix generator, a matrix generator such that*

$$A(i,j) = s((j-1)*n+i), \quad 1 \le i,j \le n. \tag{4}$$

and s is such that

$$s(i+2^{31}) = s(i), \quad \textit{for any } i \in \mathbb{N} \quad \textit{and} \quad s(i) \ne s(j), \quad \textit{for any } 1 \le i,j \le 2^{31}. \tag{5}$$

Some remarks:

1. The assumption $s(i) \ne s(j)$, for any $1 \le i,j \le 2^{31}$ is true in the case of the Linpack Benchmark matrix generator. It can be relaxed to admit more sequences s for which some elements can be identical. However this assumption makes the sufficiency proof of the theorem in §4 easier and clearer.

2. It is important to note that the matrix generated by the Linpack Benchmark matrix generator solely depends on the dimension n. The Linpack Benchmark matrix generator requires benchmarkers to use the same matrix for any block size, for any number of processors or for any grid size.

3. Moreover, since the Linpack Benchmark matrix generator possesses its own implementation of the pseudo–random generator, the computed pseudo–random numbers in the sequence s depend weakly on the computer systems. Consequently the pivot pattern of the Gaussian elimination is preserved from one computer system to the other, from one year to the other.

4. Finally, the linear congruential algorithm for the sequence s enables the matrix generator for a scalable implementation of the construction of the matrix: each process can generate their local part of the global matrix without communicating or generating the global matrix. This property is not usual among pseudo–random generators.

3 Understanding $\mathcal{S}$

Consider a large dense matrix of order $3 \cdot 10^6$ generated by the process described in Definition 2. The number of entries in this matrix is $9 \cdot 10^{12}$ which is above the pseudo–random generator period ($2^{31} \approx 2.14 \cdot 10^9$). However, despite this fact, it is fairly likely for the constructed matrix to have distinct columns and even to be well–conditioned.

On the other hand, we can easily generate a "small" matrix with identical columns. Take n=2^{16}, we have for any $i = 1,\ldots,n$:

$$A(i,2^{15}+1) = s(i+n*(j-1)) = s(i+2^{15}*n) = s(i+2^{15}*2^{16}) = s(i+2^{31}) = s(i) = A(i,1),$$

therefore the column 1 and the column $2^{15}+1$ are exactly the same. The column 2 and the column $2^{15}+2$ are exactly the same, etc. We can actually prove that $2^{16} = 65,536$ is the smallest matrix order for which a multiple of a column can happen.

Another example of $n \in \mathcal{S}$ is $n = 2^{31} = 2,147,483,648$ for which all columns of the generated matrix are the same. Our goal in this section is to build more n in $\mathcal{S}$ to have a better knowledge of this set.

If n is a multiple of $2^0 = 1$ and $n > 2^{31}$ then $n \in \mathcal{S}$. (Note that the statement "any n is multiple of $2^0 = 1$ and $n > 2^{31}$" means $n > 2^{31}$.) The reasoning is as follows. There are 2^{31} indexes from 1 to 2^{31}. Since there are at least $2^{31}+1$ elements in the first row of A (assumption $n > 2^{31}$), then, necessarily, at least one index (say k) is repeated twice in the first row of A. This is the pigeonhole principle. Therefore we have proved the existence of two columns i and j such that they both start with the k–th term of the sequence. If two columns start with the index of the sequence, they are the same (since we take the element of the column sequentially in the sequence). The three smallest numbers of this type are

$$n = 2^0*(2^{31}+1) = 2,147,483,649 \in \mathcal{S}$$
$$n = 2^0*(2^{31}+2) = 2,147,483,650 \in \mathcal{S}$$
$$n = 2^0*(2^{31}+3) = 2,147,483,651 \in \mathcal{S}$$

If n is a multiple of $2^1 = 2$ and $n > 2^{30}$ then $n \in \mathcal{S}$. If n is even ($n = 2q$), then the first row of A accesses the numbers of the sequence s using only odd indexes. There are 2^{30} odd indexes between 1 and 2^{31}. Since there are at least $2^{30}+1$ elements in the first row of A (assumption $n > 2^{30}$), then, necessarily, at least one index is repeated twice in the first row of A. This is the pigeonhole principle. The three smallest numbers of this type are:

$$n = 2^1*(2^{29}+1) = 1,073,741,826 \in \mathcal{S}$$
$$n = 2^1*(2^{29}+2) = 1,073,741,828 \in \mathcal{S}$$
$$n = 2^1*(2^{29}+3) = 1,073,741,830 \in \mathcal{S}.$$

If n is a multiple of $2^2 = 4$ and $n > 2^{29}$ then $n \in \mathcal{S}$. If n is a multiple of 4 ($n = 4q$), then the first row of A accesses the numbers of the sequence s using only $(4q+1)$–indexes. There are 2^{29} $(4q+1)$–indexes between 1 and 2^{31}. Since there are at least $2^{29}+1$ elements in the first row of A (assumption $n > 2^{29}$), then, necessarily, at least one index is repeated twice in the first row of A. This is the pigeonhole principle. The first three numbers of this type are:

$$n = 2^2*(2^{27}+1) = 536,870,916 \in \mathcal{S}$$
$$n = 2^2*(2^{27}+2) = 536,870,920 \in \mathcal{S}$$
$$n = 2^2*(2^{27}+3) = 536,870,924 \in \mathcal{S}.$$

$$\vdots$$

If n is a multiple of 2^{13} and $n > 2^{18}$ then $n \in \mathcal{S}$. This gives for example:

$$n_{12} = 2^{13}*(2^5+1) = 2^{13}*33 \quad = \quad 270,336 \in \mathcal{S}$$
$$n_{13} = 2^{13}*(2^5+2) = 2^{13}*34 \quad = \quad 278,528 \in \mathcal{S}$$
$$n_{15} = 2^{13}*(2^5+3) = 2^{13}*35 \quad = \quad 294,912 \in \mathcal{S}.$$

These three numbers correspond to entries $(3,2)$, $(3,3)$ and $(3,5)$ in Table 1.

If n is a multiple of 2^{14} and $n > 2^{17}$ then $n \in \mathcal{S}$. This gives for example:

$$\begin{aligned} n_4 = 2^{14} * (2^3 + 1) = 2^{14} * 9 &= 147,456 \in \mathcal{S} \\ n_5 = 2^{14} * (2^3 + 2) = 2^{14} * 10 &= 163,840 \in \mathcal{S} \\ n_6 = 2^{14} * (2^3 + 3) = 2^{14} * 11 &= 180,224 \in \mathcal{S}. \end{aligned}$$

These three numbers correspond to entries $(1,4)$, $(1,5)$ and $(2,1)$ in Table 1.

If n is a multiple of 2^{15} and $n > 2^{16}$ then $n \in \mathcal{S}$. This gives for example:

$$\begin{aligned} n_2 = 2^{15} * (2^1 + 1) = 2^{15} * 3 &= 98,304 \in \mathcal{S} \\ n_3 = 2^{15} * (2^1 + 2) = 2^{15} * 4 &= 131,072 \in \mathcal{S} \\ n_5 = 2^{15} * (2^1 + 3) = 2^{15} * 5 &= 163,840 \in \mathcal{S}. \end{aligned}$$

These three numbers correspond to entries $(1,2)$, $(1,3)$ and $(1,5)$ in Table 1.

If n is a multiple of 2^{16} and $n > 2^{15}$ then $n \in \mathcal{S}$.

$$\begin{aligned} n_1 = 2^{16} * (2^0 + 1) = 2^{16} * 1 &= 65,536 \in \mathcal{S} \\ n_3 = 2^{16} * (2^0 + 2) = 2^{16} * 2 &= 131,072 \in \mathcal{S} \\ n_7 = 2^{16} * (2^0 + 3) = 2^{16} * 3 &= 196,608 \in \mathcal{S}. \end{aligned}$$

These three numbers correspond to entries $(1,1)$, $(1,3)$ and $(2,2)$ in Table 1.

From this section, we understand that any n multiple of 2^k and larger than 2^{31-k} is in $\mathcal{S}$. In the next paragraph, we prove that this is indeed the only integers in $\mathcal{S}$ which provides us with a complete characterization of $\mathcal{S}$.

4 Characterization of $\mathcal{S}$

Theorem: $n \in \mathcal{S}$ if and only if the matrix of size n generated by the Linpack Benchmark matrix generator has at least two identical columns if and only if

$$n > 2^{31-k} \quad \text{where } n = 2^k \cdot q \text{ with } q \text{ odd.}$$

Proof:

$\boxed{\Leftarrow}$ Let us assume that n is a multiple of 2^k, that is to say

$$n = 2^k \cdot q, \quad 1 \leq q$$

and let us assume that

$$n > 2^{31-k}.$$

In this case, the first row of A accesses the numbers of the sequence s using only $(2^k \cdot q + 1)$–indexes. There are 2^{31-k} $(2^k \cdot q + 1)$–indexes between 1 and 2^{31}. Since there are at least $2^{31-k} + 1$ elements in the first row of A (assumption $n > 2^{31-k}$), then, necessarily, at least one index is repeated twice in the first row of A. This is the pigeonhole principle. If two columns start with the same index in the sequence, they are the same (since we take the element of the column sequentially in the sequence).

$\boxed{\Rightarrow}$ Assume that there are two identical columns i and j in the matrix generated by the Linpack Benchmark matrix generator ($i \neq j$). Without loss of generality, assume $i > j$. The fact that column i is the same as column j means that these columns have identical entries, in particular, they share the same first entry. We have

$$A(1,i) = A(1,j).$$

From this, Equation (4) implies

$$s(1 + (i-1)n) = s(1 + (j-1)n).$$

Equation (5) states that all elements in a period of length 2^{31} are different, therefore, since $i \neq j$, we necessarily have

$$1 + (i-1)n = 1 + (j-1)n + 2^{31} \cdot p, \quad 1 \leq p.$$

This implies

$$(i-j)n = 2^{31} \cdot p, \quad 1 \leq p.$$

We now use the fact that $n = 2^k \cdot q$ with q odd and get

$$(i-j) \cdot 2^k \cdot q = 2^{31} \cdot p, \quad 1 \leq p, \quad q \text{ is odd.}$$

Since q is odd, this last equality implies that 2^{31} is a divisor of $(i-j) \cdot 2^k$. This writes

$$(i-j) \cdot 2^k = 2^{31} \cdot r, \quad 1 \leq r.$$

From which, we deduce that

$$(i-j) \cdot 2^k \geq 2^{31}.$$

A upper bound for i is n, a lower bound for j is 1; therefore,

$$(n-1) \cdot 2^k \geq 2^{31}.$$

We conclude that, if a matrix of size n generated by the Linpack Benchmark matrix generator has at least two identical columns, this implies

$$n > 2^{31-k} \quad \text{where } n = 2^k \cdot q \text{ with } q \text{ odd.}$$

$\square$

5 Anomalies in Matrix Sizes Reported in the June 2008 Top500 List

Readers of this manuscript may be surprised to find three entries in the TOP 500 data from June 2008 with matrix sizes that lead to matrices with identical columns if the HPL test matrix generator is used. These three entries are given in Table 2. For example, the run for the Earth Simulator from 2002 was done with $n = 1,075,200$ which corresponds to $2^{11} \cdot 525$, therefore, the column $j = 2^{20} = 1,048,576$ would have been a repeat of the first under our assumptions. The benchmark run on the Earth Similator in 2002 was done with an older version of the test harness. This test harness predates the HPL test harness and uses another matrix generator than the one provided by HPL. Today we require the HPL test harness to be used in the benchmark run.

Rank	Site	Manufacturer	Year	NMax
16	Information Technology Center, The University of Tokyo	Hitachi	2008	1,433,600
49	The Earth Simulator Center	NEC	2002	1,075,200
88	Cardiff University - ARCCA	Bull SA	2008	634,880

Table 2: The three entries in the TOP500 June 2008 list with suspicious n.

6 How to fix the problem

Between 1 and $1 \cdot 10^6$, there are 49 matrix sizes in $\mathcal{S}$ (see Table 1). Between 1 and $3 \cdot 10^6$, there are $1,546$ matrix sizes in $\mathcal{S}$ (see Appendix B). Therefore, for this order of matrix size, there is a good chance to choose a matrix size that is not in $\mathcal{S}$. Unfortunately benchmarkers tend to pick multiples of high power of 2 for their matrix sizes which increases the likelihood of picking an $n \in \mathcal{S}$.

1. The obvious recommendation is to choose any n as long as it is odd. In the odd case if $n < 2^{31} \approx 4 \cdot 10^9$, then $n \notin \mathcal{S}$.

2. A check can be added at the beginning of the execution of the Linpack Benchmark matrix generator. The C-code looks as follows:
```
long long int m,n;
int i,k,t,s;
s = 31;
m=n; k=0; while (m%2==0) {k++; m=m/2;}
m=1; t=0; while (m<=n) {t++; m=m*2;}
if (t+k>s) i = 1; else i = 0;
```
   n is the matrix size, 2^s is period of the pseudo–random number generator ($s = 31$ in our case) and i is the output flag. If $i = 1$, then $n \in \mathcal{S}$. If $i = 0$, then $n \notin \mathcal{S}$. (The check could also consist of looking over the data given in Appendix B).

3. If $n \in \mathcal{S}$, one can simply pad the matrix with an extra line. This can be easily done in the HPL code `HPL_pdmatgen.c` by changing the variable `jump3` from `M` to `M+1` whenever $n \in \mathcal{S}$.

4. Another possibility is to increase the period of the pseudo–random generator used. For example, if the pseudo–random generator had a period of 2^{64} and if $n \le 2^{32}$, then, assuming $(i \ne j \Rightarrow s(i) \ne s(j))$, entries would never repeat.

We are planning to correct the problem.

Acknowledgements

The authors would like to thank Piotr Luszczek and Antoine Petitet for their valuable comments on HPL, and Asim Yarkhan for one pertinent observation.

References

[1] D. E. Knuth. *The Art of Computer Programming*, volume 2. Addison–Wesley, third edition, 1997.

[2] http://www.netlib.org/benchmark/hpl/.

[3] http://www.top500.org/.

A The $1,564$ matrix sizes of n from 1 to $3,000,000$ for which the Linpack Benchmark matrix generator will construct a matrix with identical columns

65,536	98,304	131,072	147,456	163,840	180,224	196,608	212,992	229,376	245,760
262,144	270,336	278,528	286,720	294,912	303,104	311,296	319,488	327,680	335,872
344,064	352,256	360,448	368,640	376,832	385,024	393,216	401,408	409,600	417,792
425,984	434,176	442,368	450,560	458,752	466,944	475,136	483,328	491,520	499,712
507,904	516,096	524,288	528,384	532,480	536,576	540,672	544,768	548,864	552,960
557,056	561,152	565,248	569,344	573,440	577,536	581,632	585,728	589,824	593,920
598,016	602,112	606,208	610,304	614,400	618,496	622,592	626,688	630,784	634,880
638,976	643,072	647,168	651,264	655,360	659,456	663,552	667,648	671,744	675,840
679,936	684,032	688,128	692,224	696,320	700,416	704,512	708,608	712,704	716,800
720,896	724,992	729,088	733,184	737,280	741,376	745,472	749,568	753,664	757,760
761,856	765,952	770,048	774,144	778,240	782,336	786,432	790,528	794,624	798,720
802,816	806,912	811,008	815,104	819,200	823,296	827,392	831,488	835,584	839,680
843,776	847,872	851,968	856,064	860,160	864,256	868,352	872,448	876,544	880,640
884,736	888,832	892,928	897,024	901,120	905,216	909,312	913,408	917,504	921,600
925,696	929,792	933,888	937,984	942,080	946,176	950,272	954,368	958,464	962,560
966,656	970,752	974,848	978,944	983,040	987,136	991,232	995,328	999,424	1,003,520
1,007,616	1,011,712	1,015,808	1,019,904	1,024,000	1,028,096	1,032,192	1,036,288	1,040,384	1,044,480
1,048,576	1,050,624	1,052,672	1,054,720	1,056,768	1,058,816	1,060,864	1,062,912	1,064,960	1,067,008
1,069,056	1,071,104	1,073,152	1,075,200	1,077,248	1,079,296	1,081,344	1,083,392	1,085,440	1,087,488
1,089,536	1,091,584	1,093,632	1,095,680	1,097,728	1,099,776	1,101,824	1,103,872	1,105,920	1,107,968
1,110,016	1,112,064	1,114,112	1,116,160	1,118,208	1,120,256	1,122,304	1,124,352	1,126,400	1,128,448
1,130,496	1,132,544	1,134,592	1,136,640	1,138,688	1,140,736	1,142,784	1,144,832	1,146,880	1,148,928
1,150,976	1,153,024	1,155,072	1,157,120	1,159,168	1,161,216	1,163,264	1,165,312	1,167,360	1,169,408
1,171,456	1,173,504	1,175,552	1,177,600	1,179,648	1,181,696	1,183,744	1,185,792	1,187,840	1,189,888
1,191,936	1,193,984	1,196,032	1,198,080	1,200,128	1,202,176	1,204,224	1,206,272	1,208,320	1,210,368
1,212,416	1,214,464	1,216,512	1,218,560	1,220,608	1,222,656	1,224,704	1,226,752	1,228,800	1,230,848
1,232,896	1,234,944	1,236,992	1,239,040	1,241,088	1,243,136	1,245,184	1,247,232	1,249,280	1,251,328
1,253,376	1,255,424	1,257,472	1,259,520	1,261,568	1,263,616	1,265,664	1,267,712	1,269,760	1,271,808
1,273,856	1,275,904	1,277,952	1,280,000	1,282,048	1,284,096	1,286,144	1,288,192	1,290,240	1,292,288
1,294,336	1,296,384	1,298,432	1,300,480	1,302,528	1,304,576	1,306,624	1,308,672	1,310,720	1,312,768
1,314,816	1,316,864	1,318,912	1,320,960	1,323,008	1,325,056	1,327,104	1,329,152	1,331,200	1,333,248
1,335,296	1,337,344	1,339,392	1,341,440	1,343,488	1,345,536	1,347,584	1,349,632	1,351,680	1,353,728
1,355,776	1,357,824	1,359,872	1,361,920	1,363,968	1,366,016	1,368,064	1,370,112	1,372,160	1,374,208
1,376,256	1,378,304	1,380,352	1,382,400	1,384,448	1,386,496	1,388,544	1,390,592	1,392,640	1,394,688
1,396,736	1,398,784	1,400,832	1,402,880	1,404,928	1,406,976	1,409,024	1,411,072	1,413,120	1,415,168
1,417,216	1,419,264	1,421,312	1,423,360	1,425,408	1,427,456	1,429,504	1,431,552	1,433,600	1,435,648
1,437,696	1,439,744	1,441,792	1,443,840	1,445,888	1,447,936	1,449,984	1,452,032	1,454,080	1,456,128
1,458,176	1,460,224	1,462,272	1,464,320	1,466,368	1,468,416	1,470,464	1,472,512	1,474,560	1,476,608
1,478,656	1,480,704	1,482,752	1,484,800	1,486,848	1,488,896	1,490,944	1,492,992	1,495,040	1,497,088
1,499,136	1,501,184	1,503,232	1,505,280	1,507,328	1,509,376	1,511,424	1,513,472	1,515,520	1,517,568
1,519,616	1,521,664	1,523,712	1,525,760	1,527,808	1,529,856	1,531,904	1,533,952	1,536,000	1,538,048
1,540,096	1,542,144	1,544,192	1,546,240	1,548,288	1,550,336	1,552,384	1,554,432	1,556,480	1,558,528
1,560,576	1,562,624	1,564,672	1,566,720	1,568,768	1,570,816	1,572,864	1,574,912	1,576,960	1,579,008
1,581,056	1,583,104	1,585,152	1,587,200	1,589,248	1,591,296	1,593,344	1,595,392	1,597,440	1,599,488
1,601,536	1,603,584	1,605,632	1,607,680	1,609,728	1,611,776	1,613,824	1,615,872	1,617,920	1,619,968
1,622,016	1,624,064	1,626,112	1,628,160	1,630,208	1,632,256	1,634,304	1,636,352	1,638,400	1,640,448
1,642,496	1,644,544	1,646,592	1,648,640	1,650,688	1,652,736	1,654,784	1,656,832	1,658,880	1,660,928
1,662,976	1,665,024	1,667,072	1,669,120	1,671,168	1,673,216	1,675,264	1,677,312	1,679,360	1,681,408
1,683,456	1,685,504	1,687,552	1,689,600	1,691,648	1,693,696	1,695,744	1,697,792	1,699,840	1,701,888
1,703,936	1,705,984	1,708,032	1,710,080	1,712,128	1,714,176	1,716,224	1,718,272	1,720,320	1,722,368

1,724,416	1,726,464	1,728,512	1,730,560	1,732,608	1,734,656	1,736,704	1,738,752	1,740,800	1,742,848
1,744,896	1,746,944	1,748,992	1,751,040	1,753,088	1,755,136	1,757,184	1,759,232	1,761,280	1,763,328
1,765,376	1,767,424	1,769,472	1,771,520	1,773,568	1,775,616	1,777,664	1,779,712	1,781,760	1,783,808
1,785,856	1,787,904	1,789,952	1,792,000	1,794,048	1,796,096	1,798,144	1,800,192	1,802,240	1,804,288
1,806,336	1,808,384	1,810,432	1,812,480	1,814,528	1,816,576	1,818,624	1,820,672	1,822,720	1,824,768
1,826,816	1,828,864	1,830,912	1,832,960	1,835,008	1,837,056	1,839,104	1,841,152	1,843,200	1,845,248
1,847,296	1,849,344	1,851,392	1,853,440	1,855,488	1,857,536	1,859,584	1,861,632	1,863,680	1,865,728
1,867,776	1,869,824	1,871,872	1,873,920	1,875,968	1,878,016	1,880,064	1,882,112	1,884,160	1,886,208
1,888,256	1,890,304	1,892,352	1,894,400	1,896,448	1,898,496	1,900,544	1,902,592	1,904,640	1,906,688
1,908,736	1,910,784	1,912,832	1,914,880	1,916,928	1,918,976	1,921,024	1,923,072	1,925,120	1,927,168
1,929,216	1,931,264	1,933,312	1,935,360	1,937,408	1,939,456	1,941,504	1,943,552	1,945,600	1,947,648
1,949,696	1,951,744	1,953,792	1,955,840	1,957,888	1,959,936	1,961,984	1,964,032	1,966,080	1,968,128
1,970,176	1,972,224	1,974,272	1,976,320	1,978,368	1,980,416	1,982,464	1,984,512	1,986,560	1,988,608
1,990,656	1,992,704	1,994,752	1,996,800	1,998,848	2,000,896	2,002,944	2,004,992	2,007,040	2,009,088
2,011,136	2,013,184	2,015,232	2,017,280	2,019,328	2,021,376	2,023,424	2,025,472	2,027,520	2,029,568
2,031,616	2,033,664	2,035,712	2,037,760	2,039,808	2,041,856	2,043,904	2,045,952	2,048,000	2,050,048
2,052,096	2,054,144	2,056,192	2,058,240	2,060,288	2,062,336	2,064,384	2,066,432	2,068,480	2,070,528
2,072,576	2,074,624	2,076,672	2,078,720	2,080,768	2,082,816	2,084,864	2,086,912	2,088,960	2,091,008
2,093,056	2,095,104	2,097,152	2,098,176	2,099,200	2,100,224	2,101,248	2,102,272	2,103,296	2,104,320
2,105,344	2,106,368	2,107,392	2,108,416	2,109,440	2,110,464	2,111,488	2,112,512	2,113,536	2,114,560
2,115,584	2,116,608	2,117,632	2,118,656	2,119,680	2,120,704	2,121,728	2,122,752	2,123,776	2,124,800
2,125,824	2,126,848	2,127,872	2,128,896	2,129,920	2,130,944	2,131,968	2,132,992	2,134,016	2,135,040
2,136,064	2,137,088	2,138,112	2,139,136	2,140,160	2,141,184	2,142,208	2,143,232	2,144,256	2,145,280
2,146,304	2,147,328	2,148,352	2,149,376	2,150,400	2,151,424	2,152,448	2,153,472	2,154,496	2,155,520
2,156,544	2,157,568	2,158,592	2,159,616	2,160,640	2,161,664	2,162,688	2,163,712	2,164,736	2,165,760
2,166,784	2,167,808	2,168,832	2,169,856	2,170,880	2,171,904	2,172,928	2,173,952	2,174,976	2,176,000
2,177,024	2,178,048	2,179,072	2,180,096	2,181,120	2,182,144	2,183,168	2,184,192	2,185,216	2,186,240
2,187,264	2,188,288	2,189,312	2,190,336	2,191,360	2,192,384	2,193,408	2,194,432	2,195,456	2,196,480
2,197,504	2,198,528	2,199,552	2,200,576	2,201,600	2,202,624	2,203,648	2,204,672	2,205,696	2,206,720
2,207,744	2,208,768	2,209,792	2,210,816	2,211,840	2,212,864	2,213,888	2,214,912	2,215,936	2,216,960
2,217,984	2,219,008	2,220,032	2,221,056	2,222,080	2,223,104	2,224,128	2,225,152	2,226,176	2,227,200
2,228,224	2,229,248	2,230,272	2,231,296	2,232,320	2,233,344	2,234,368	2,235,392	2,236,416	2,237,440
2,238,464	2,239,488	2,240,512	2,241,536	2,242,560	2,243,584	2,244,608	2,245,632	2,246,656	2,247,680
2,248,704	2,249,728	2,250,752	2,251,776	2,252,800	2,253,824	2,254,848	2,255,872	2,256,896	2,257,920
2,258,944	2,259,968	2,260,992	2,262,016	2,263,040	2,264,064	2,265,088	2,266,112	2,267,136	2,268,160
2,269,184	2,270,208	2,271,232	2,272,256	2,273,280	2,274,304	2,275,328	2,276,352	2,277,376	2,278,400
2,279,424	2,280,448	2,281,472	2,282,496	2,283,520	2,284,544	2,285,568	2,286,592	2,287,616	2,288,640
2,289,664	2,290,688	2,291,712	2,292,736	2,293,760	2,294,784	2,295,808	2,296,832	2,297,856	2,298,880
2,299,904	2,300,928	2,301,952	2,302,976	2,304,000	2,305,024	2,306,048	2,307,072	2,308,096	2,309,120
2,310,144	2,311,168	2,312,192	2,313,216	2,314,240	2,315,264	2,316,288	2,317,312	2,318,336	2,319,360
2,320,384	2,321,408	2,322,432	2,323,456	2,324,480	2,325,504	2,326,528	2,327,552	2,328,576	2,329,600
2,330,624	2,331,648	2,332,672	2,333,696	2,334,720	2,335,744	2,336,768	2,337,792	2,338,816	2,339,840
2,340,864	2,341,888	2,342,912	2,343,936	2,344,960	2,345,984	2,347,008	2,348,032	2,349,056	2,350,080
2,351,104	2,352,128	2,353,152	2,354,176	2,355,200	2,356,224	2,357,248	2,358,272	2,359,296	2,360,320
2,361,344	2,362,368	2,363,392	2,364,416	2,365,440	2,366,464	2,367,488	2,368,512	2,369,536	2,370,560
2,371,584	2,372,608	2,373,632	2,374,656	2,375,680	2,376,704	2,377,728	2,378,752	2,379,776	2,380,800
2,381,824	2,382,848	2,383,872	2,384,896	2,385,920	2,386,944	2,387,968	2,388,992	2,390,016	2,391,040
2,392,064	2,393,088	2,394,112	2,395,136	2,396,160	2,397,184	2,398,208	2,399,232	2,400,256	2,401,280
2,402,304	2,403,328	2,404,352	2,405,376	2,406,400	2,407,424	2,408,448	2,409,472	2,410,496	2,411,520
2,412,544	2,413,568	2,414,592	2,415,616	2,416,640	2,417,664	2,418,688	2,419,712	2,420,736	2,421,760
2,422,784	2,423,808	2,424,832	2,425,856	2,426,880	2,427,904	2,428,928	2,429,952	2,430,976	2,432,000
2,433,024	2,434,048	2,435,072	2,436,096	2,437,120	2,438,144	2,439,168	2,440,192	2,441,216	2,442,240
2,443,264	2,444,288	2,445,312	2,446,336	2,447,360	2,448,384	2,449,408	2,450,432	2,451,456	2,452,480
2,453,504	2,454,528	2,455,552	2,456,576	2,457,600	2,458,624	2,459,648	2,460,672	2,461,696	2,462,720
2,463,744	2,464,768	2,465,792	2,466,816	2,467,840	2,468,864	2,469,888	2,470,912	2,471,936	2,472,960

2,473,984	2,475,008	2,476,032	2,477,056	2,478,080	2,479,104	2,480,128	2,481,152	2,482,176	2,483,200
2,484,224	2,485,248	2,486,272	2,487,296	2,488,320	2,489,344	2,490,368	2,491,392	2,492,416	2,493,440
2,494,464	2,495,488	2,496,512	2,497,536	2,498,560	2,499,584	2,500,608	2,501,632	2,502,656	2,503,680
2,504,704	2,505,728	2,506,752	2,507,776	2,508,800	2,509,824	2,510,848	2,511,872	2,512,896	2,513,920
2,514,944	2,515,968	2,516,992	2,518,016	2,519,040	2,520,064	2,521,088	2,522,112	2,523,136	2,524,160
2,525,184	2,526,208	2,527,232	2,528,256	2,529,280	2,530,304	2,531,328	2,532,352	2,533,376	2,534,400
2,535,424	2,536,448	2,537,472	2,538,496	2,539,520	2,540,544	2,541,568	2,542,592	2,543,616	2,544,640
2,545,664	2,546,688	2,547,712	2,548,736	2,549,760	2,550,784	2,551,808	2,552,832	2,553,856	2,554,880
2,555,904	2,556,928	2,557,952	2,558,976	2,560,000	2,561,024	2,562,048	2,563,072	2,564,096	2,565,120
2,566,144	2,567,168	2,568,192	2,569,216	2,570,240	2,571,264	2,572,288	2,573,312	2,574,336	2,575,360
2,576,384	2,577,408	2,578,432	2,579,456	2,580,480	2,581,504	2,582,528	2,583,552	2,584,576	2,585,600
2,586,624	2,587,648	2,588,672	2,589,696	2,590,720	2,591,744	2,592,768	2,593,792	2,594,816	2,595,840
2,596,864	2,597,888	2,598,912	2,599,936	2,600,960	2,601,984	2,603,008	2,604,032	2,605,056	2,606,080
2,607,104	2,608,128	2,609,152	2,610,176	2,611,200	2,612,224	2,613,248	2,614,272	2,615,296	2,616,320
2,617,344	2,618,368	2,619,392	2,620,416	2,621,440	2,622,464	2,623,488	2,624,512	2,625,536	2,626,560
2,627,584	2,628,608	2,629,632	2,630,656	2,631,680	2,632,704	2,633,728	2,634,752	2,635,776	2,636,800
2,637,824	2,638,848	2,639,872	2,640,896	2,641,920	2,642,944	2,643,968	2,644,992	2,646,016	2,647,040
2,648,064	2,649,088	2,650,112	2,651,136	2,652,160	2,653,184	2,654,208	2,655,232	2,656,256	2,657,280
2,658,304	2,659,328	2,660,352	2,661,376	2,662,400	2,663,424	2,664,448	2,665,472	2,666,496	2,667,520
2,668,544	2,669,568	2,670,592	2,671,616	2,672,640	2,673,664	2,674,688	2,675,712	2,676,736	2,677,760
2,678,784	2,679,808	2,680,832	2,681,856	2,682,880	2,683,904	2,684,928	2,685,952	2,686,976	2,688,000
2,689,024	2,690,048	2,691,072	2,692,096	2,693,120	2,694,144	2,695,168	2,696,192	2,697,216	2,698,240
2,699,264	2,700,288	2,701,312	2,702,336	2,703,360	2,704,384	2,705,408	2,706,432	2,707,456	2,708,480
2,709,504	2,710,528	2,711,552	2,712,576	2,713,600	2,714,624	2,715,648	2,716,672	2,717,696	2,718,720
2,719,744	2,720,768	2,721,792	2,722,816	2,723,840	2,724,864	2,725,888	2,726,912	2,727,936	2,728,960
2,729,984	2,731,008	2,732,032	2,733,056	2,734,080	2,735,104	2,736,128	2,737,152	2,738,176	2,739,200
2,740,224	2,741,248	2,742,272	2,743,296	2,744,320	2,745,344	2,746,368	2,747,392	2,748,416	2,749,440
2,750,464	2,751,488	2,752,512	2,753,536	2,754,560	2,755,584	2,756,608	2,757,632	2,758,656	2,759,680
2,760,704	2,761,728	2,762,752	2,763,776	2,764,800	2,765,824	2,766,848	2,767,872	2,768,896	2,769,920
2,770,944	2,771,968	2,772,992	2,774,016	2,775,040	2,776,064	2,777,088	2,778,112	2,779,136	2,780,160
2,781,184	2,782,208	2,783,232	2,784,256	2,785,280	2,786,304	2,787,328	2,788,352	2,789,376	2,790,400
2,791,424	2,792,448	2,793,472	2,794,496	2,795,520	2,796,544	2,797,568	2,798,592	2,799,616	2,800,640
2,801,664	2,802,688	2,803,712	2,804,736	2,805,760	2,806,784	2,807,808	2,808,832	2,809,856	2,810,880
2,811,904	2,812,928	2,813,952	2,814,976	2,816,000	2,817,024	2,818,048	2,819,072	2,820,096	2,821,120
2,822,144	2,823,168	2,824,192	2,825,216	2,826,240	2,827,264	2,828,288	2,829,312	2,830,336	2,831,360
2,832,384	2,833,408	2,834,432	2,835,456	2,836,480	2,837,504	2,838,528	2,839,552	2,840,576	2,841,600
2,842,624	2,843,648	2,844,672	2,845,696	2,846,720	2,847,744	2,848,768	2,849,792	2,850,816	2,851,840
2,852,864	2,853,888	2,854,912	2,855,936	2,856,960	2,857,984	2,859,008	2,860,032	2,861,056	2,862,080
2,863,104	2,864,128	2,865,152	2,866,176	2,867,200	2,868,224	2,869,248	2,870,272	2,871,296	2,872,320
2,873,344	2,874,368	2,875,392	2,876,416	2,877,440	2,878,464	2,879,488	2,880,512	2,881,536	2,882,560
2,883,584	2,884,608	2,885,632	2,886,656	2,887,680	2,888,704	2,889,728	2,890,752	2,891,776	2,892,800
2,893,824	2,894,848	2,895,872	2,896,896	2,897,920	2,898,944	2,899,968	2,900,992	2,902,016	2,903,040
2,904,064	2,905,088	2,906,112	2,907,136	2,908,160	2,909,184	2,910,208	2,911,232	2,912,256	2,913,280
2,914,304	2,915,328	2,916,352	2,917,376	2,918,400	2,919,424	2,920,448	2,921,472	2,922,496	2,923,520
2,924,544	2,925,568	2,926,592	2,927,616	2,928,640	2,929,664	2,930,688	2,931,712	2,932,736	2,933,760
2,934,784	2,935,808	2,936,832	2,937,856	2,938,880	2,939,904	2,940,928	2,941,952	2,942,976	2,944,000
2,945,024	2,946,048	2,947,072	2,948,096	2,949,120	2,950,144	2,951,168	2,952,192	2,953,216	2,954,240
2,955,264	2,956,288	2,957,312	2,958,336	2,959,360	2,960,384	2,961,408	2,962,432	2,963,456	2,964,480
2,965,504	2,966,528	2,967,552	2,968,576	2,969,600	2,970,624	2,971,648	2,972,672	2,973,696	2,974,720
2,975,744	2,976,768	2,977,792	2,978,816	2,979,840	2,980,864	2,981,888	2,982,912	2,983,936	2,984,960
2,985,984	2,987,008	2,988,032	2,989,056	2,990,080	2,991,104	2,992,128	2,993,152	2,994,176	2,995,200
2,996,224	2,997,248	2,998,272	2,999,296						