

Clemson University, Conference October 2004



1/31

# Linear Dependency and the Quadratic Sieve

Kim Bowman and Zach Cochran

Clemson University and University of Georgia

and

Neil Calkin, Tim Flowers, Kevin James and Shannon Purvis





# Introduction to Factoring

Fermat's method: difference of two squares

$$a^2 \equiv b^2 \pmod{n} \leftrightarrow n \mid (a^2 - b^2) = (a + b)(a - b)$$

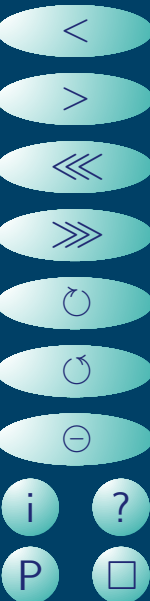
If  $a$  is not  $\equiv (\pm b) \pmod{n}$ , then  $n \nmid (a + b)$  and  $n \nmid (a - b)$

So,  $n$  must have a common factor in both.

Let  $g = \text{GCD}(a - b, n)$ .

Now,  $g = 1 \rightarrow n \mid (a + b)$  and  $g = n \rightarrow n \mid (a - b)$

So, we know that  $1 < g < n$ , which means there exists a nontrivial factor of  $n$ .





# A Brief Overview of the Quadratic Sieve

The Quadratic Sieve, or QS, is an algorithm used to factor some large integer  $n$ .

- Collect some set of primes  $P$ , where  $P = \{p_1, p_2, p_3, \dots, p_k\}$  are the primes that are  $\leq B$  for some bound  $B$ .
- We know that one-half of the primes can be thrown out due to some technical math. The remaining subset is called the factor base.
- Take an interval  $R = \{\lceil \sqrt{n} \rceil, \dots, \lceil \sqrt{n} + L - 1 \rceil\}$ , a subinterval of  $(\sqrt{n}, \sqrt{2n})$ , where  $n$  is the number you want to factor.





# A Brief Overview of the Quadratic Sieve

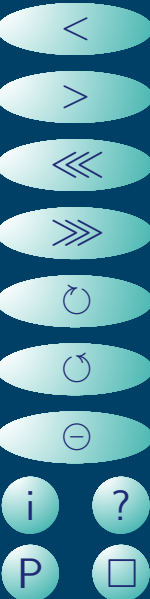
- Compute  $f(r) = r^2 - n$  for each  $r_i$  in the interval and place these values in an array.
- Divide each  $f(r_i)$  value in the array by each  $p_j$  as many times as possible. This is known as the sieving process.
- Find and collect all the entries in the array that equal one after the sieving process is complete. We'll call the elements of this subset,  $R' = \{r_1, r_2, r_3, \dots, r_m\}$ , that factor completely over all primes  $p_i \leq B$  *B-smooth*.





# A Brief Overview of the Quadratic Sieve

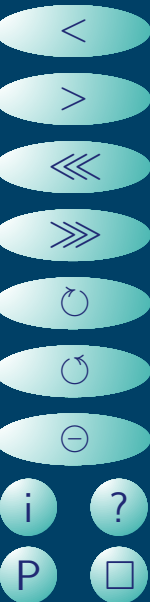
- After writing out the prime factorizations of these  $B$ -smooth numbers, take the exponents of each prime modulo two and put them in a binary matrix  $A$  where each row represents a  $B$ -smooth number and each column represents a different prime in the factor base.
- Perform Gaussian elimination on the matrix to find a linear dependency.
- We know each  $r_i^2$  is a square modulo  $n$ ; let the product of the elements in  $R'$  be  $a^2$ . The product of these  $f(r_i)$ 's is a square modulo  $n$ , call this product  $b^2$ .
- Since  $a^2 \equiv b^2 \pmod{n}$ , we compute  $GCD(n, a - b)$  to find a factor.





# Limitations of the QS

As it stands, the quadratic sieve is the second fastest general purpose factoring method. Despite this, it is still a time-consuming algorithm. The lengthiest part of the algorithm is the sieving process, but since this part of the code can be parallelized, the running time of the linear algebra becomes more important.

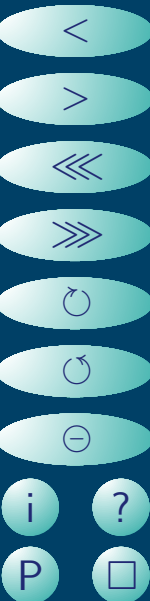




# Limitations of the QS

"As the numbers we try to factor get larger, the matrix state of QS (...) looms larger. The unfavorable bound of Gaussian elimination ruins our overall complexity estimates, which assume that the matrix stage is not a bottleneck. In addition, the awkwardness of dealing with huge matrices seems to require large and expensive computers, computers for which it is not easy to get large blocks of time."

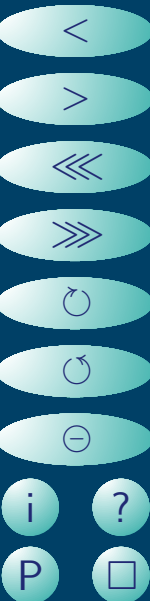
*Richard Crandell and Carl Pomerance Prime Numbers A Computational Perspective*





# Limitations of the QS

As it stands now, when there are  $k$  primes in the factor base,  $k + 1$   $B$ -smooth numbers are found by the sieving process to ensure a dependency. So, if there are  $k$  primes in the factor base and we sieve until we have  $k + 1$   $B$ -smooth numbers, then we have a  $[(k + 1) \times k]$  matrix  $A$  which is clearly row dependent. Any significant reduction in the size of  $A$  would speed up the linear algebra significantly.





# Limitations of the QS

How could we do better?

Our research focused on how many rows of  $A$  are needed to obtain a linear dependency. Such a decrease should allow us to shorten the interval we must sieve. If we do not need as many as  $k + 1$  rows in  $A$  then:

(a) We don't need to sieve as long.

(b) Linear algebra is faster.



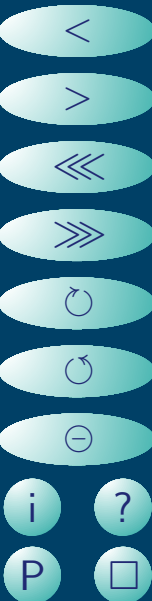
# Approaches

Let's consider a probabilistic model:

Since the entries of the exponent vectors from the sieve are either one or zero, we wrote a computer program that would generate random binary vectors, where each entry in the vector has a one with a probability designed to model the probabilities produced by the QS. The program would generate these vectors and perform Gaussian elimination until a dependency was found. The question that remained was what model should be used to generate the entries of these vectors.



10/31





# Constant Weight Vectors

Our first approach: Since the number  $n$  typically has about  $\log\log n$  prime factors, we might consider vectors in  $\mathbb{F}_2^k$  of fixed weight  $w = \log\log(n)$ . We concentrated on weight three vectors. After multiple tests were run in several different dimensions and data was collected, we found that we needed a number of vectors totaling roughly about 92 to 93% of the dimension to have a high probability that a dependency would be found.

This is in accordance with a theorem of Calkin, that essentially,  $k(1 - \frac{e^{-w}}{\log(2)})$  is a lower bound for dependency. We can see easily that  $k(1 - e^{-w})$  is (essentially) an upper bound for dependency; our results suggest that the threshold lies close to the lower bound.



# What We've Done



12/31

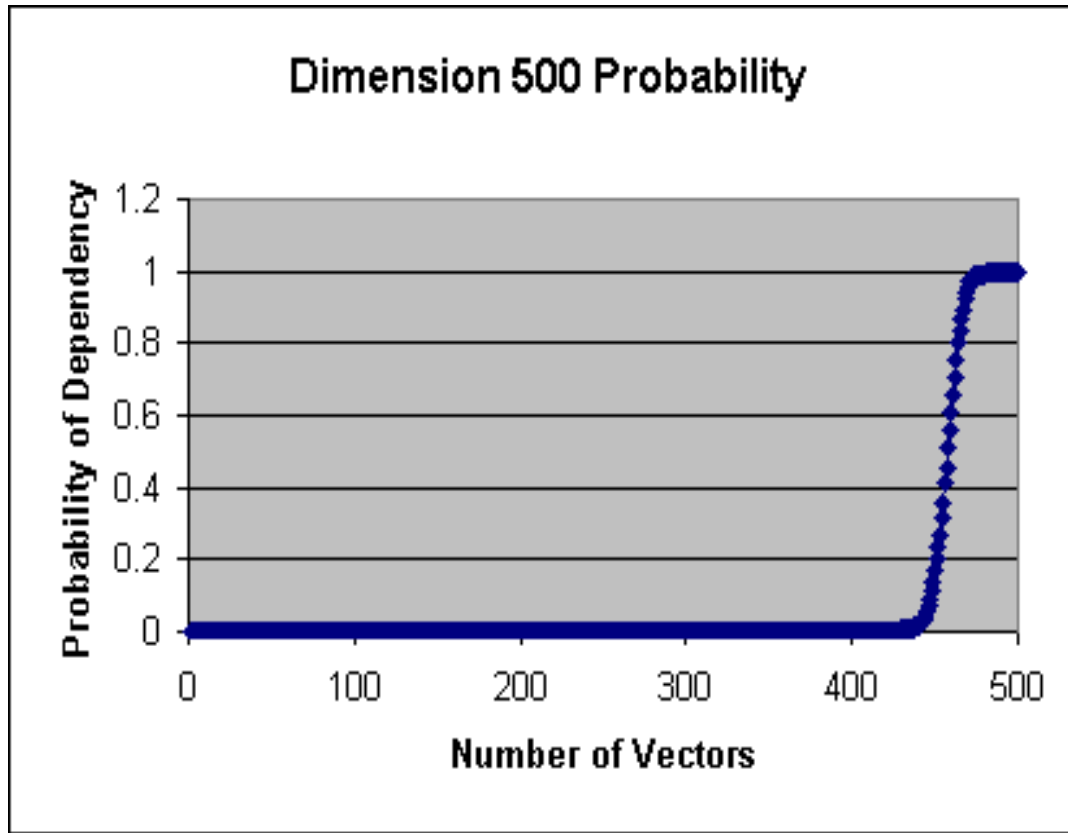
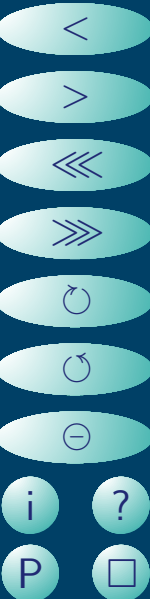


Figure 1: Dim 500 Wt 3 Probability Distribution Graph



# What We've Done

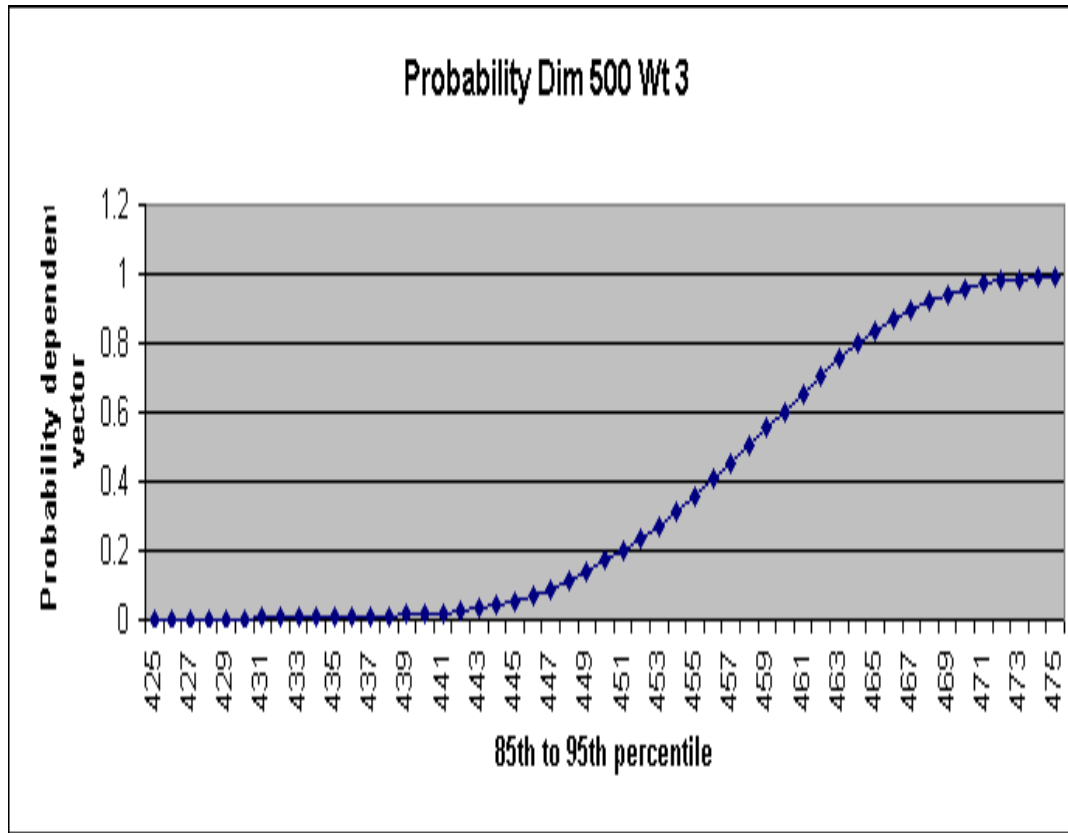
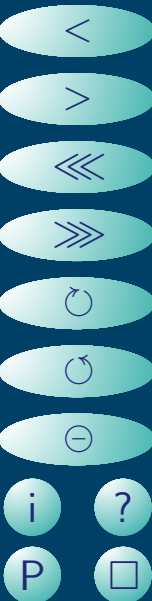


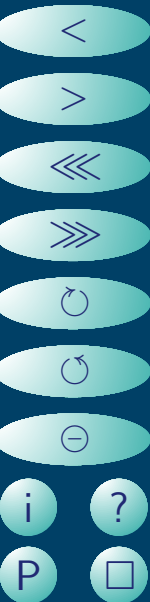
Figure 2: Dim 500 Wt 3 Probability Distribution Graph





# Is there a better model?

The next natural progression of thought leads to the question, "Are constant weight vectors a realistic model?"

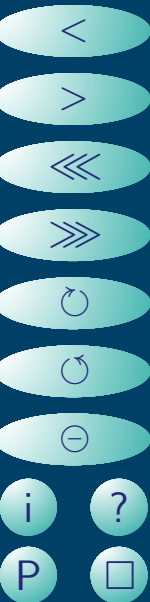




# Is there a better model?

The next natural progression of thought leads to the question, "Are constant weight vectors a realistic model?"

NO!





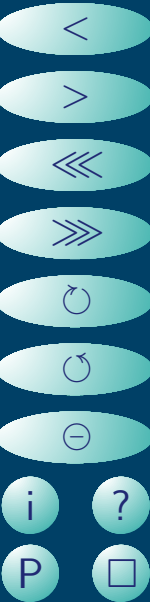
# Is there a better model?

The next natural progression of thought leads to the question, "Are constant weight vectors a realistic model?"

NO!

Why?

Where does this model go wrong?





# Is there a better model?

The next natural progression of thought leads to the question, "Are constant weight vectors a realistic model?"

NO!

Why?

Where does this model go wrong?

Small primes are too rare and big primes are too common. By uniformly picking vectors of constant weight, you are assigning the same probability of getting a one in each column. That is equivalent to saying that a number  $n$  is equally likely to be divisible by two as it is to be divisible by 7919.

How do we fix the model? A better representation of the random vectors would be to have different probabilities for each column.

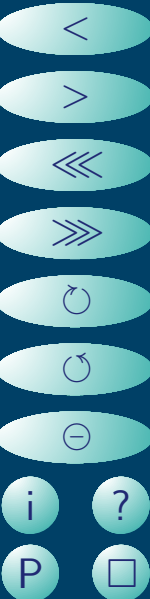




# Independent Probability Vectors

Let  $p_i$  represent the prime number associated with the  $i$ th column. Let's look at the probability  $\alpha_i$  that an odd power of  $p_i$  divides  $n$ .

$$\begin{aligned}\alpha_i &= \frac{1}{p_i} - \frac{1}{p_i^2} + \frac{1}{p_i^3} - \frac{1}{p_i^4} + \dots \\ &= \frac{1}{p_i} \left( 1 + \left( \frac{-1}{p_i} \right) + \left( \frac{-1}{p_i} \right)^2 + \dots \right) \\ &= \frac{1}{p_i} \left( \frac{1}{1 - \frac{-1}{p_i}} \right) = \frac{1}{p_i} \left( \frac{1}{1 + \frac{1}{p_i}} \right) \\ &= \frac{1}{p_i + 1}\end{aligned}$$

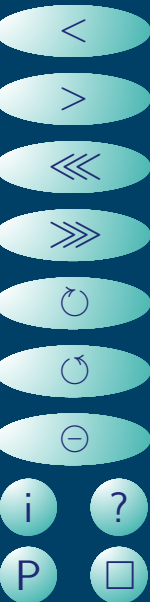


# Independent Probability Vectors

With this change in our model, dependencies occur much earlier. So far, looking at dimensions up through 25,000, we find dependencies within the first 100 rows instead of at 93% of the dimension.



19/31



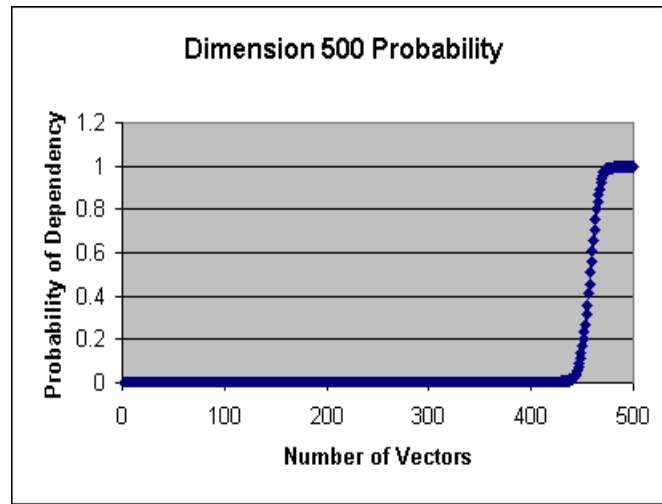


Figure 3: Dim 500 Wt 3 Probability Distribution Graph

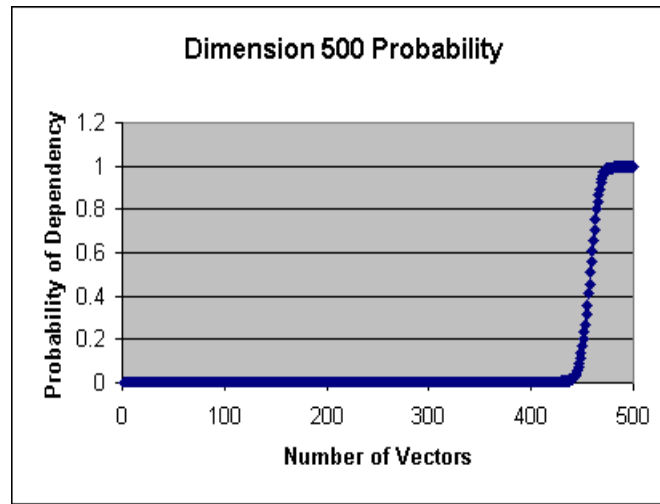


Figure 4: Dim 500 Wt 3 Probability Distribution Graph

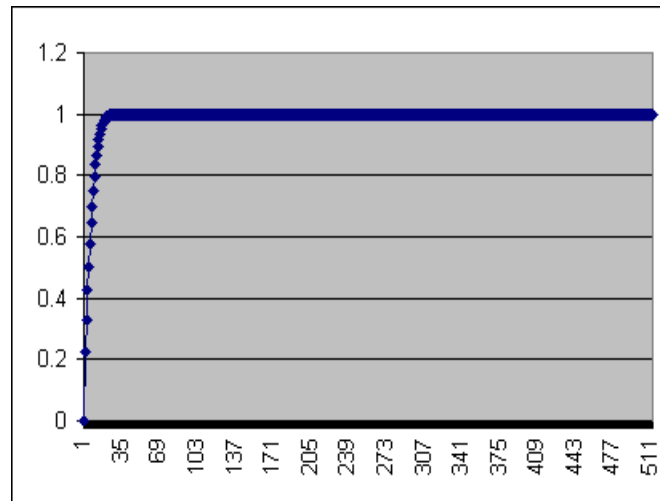


Figure 5: Probability of Dependency in Dimension 512

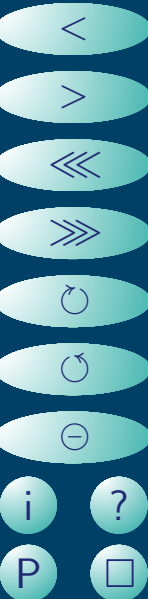


# An Even Better Model

Extensive calculations borne out by extensive computations suggest that a better model is:

$$\alpha_i = \frac{1}{p_i^{1-\delta} + 1} \quad \text{where } \delta = \frac{\log u}{\log k}$$

$$\text{where } u = \frac{\log n}{\log k}$$





# When is a set of binary vectors linearly dependent?

1. Do Gaussian elimination: Problem, slow
2. Trivial Dependency (TD): more rows than columns
3. Almost Trivial Dependency (ATD): more rows than non-empty columns
4. Nearly Trivial Dependency (NTD): continuing research (columns with 1 one)
5. Somewhat Trivial Dependency (STD): research to come (column with exactly 2 one's)



# Almost Trivial Dependencies

$$\begin{array}{ccccccc} & \downarrow & & \downarrow & & \downarrow & & \\ \left( \begin{array}{cccccccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array} \right) & \rightarrow & \left( \begin{array}{ccccc} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{array} \right)\end{array}$$



# Nearly Trivial Dependencies

$$\begin{array}{cccccccc} & \downarrow & & \downarrow & & \downarrow & \downarrow & \\ \rightarrow & \left( \begin{array}{cccccccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array} \right) & \rightarrow & \left( \begin{array}{cccc} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{array} \right) \end{array}$$



# The Expected Number of Empty Columns

The probability that the  $i$ th column has no ones is:

$$P(\text{no 1's}) = (1 - \alpha_i)^l$$

where  $l$  is the number of vectors, or rows in the matrix.

Thus, the expected number of empty columns is:

$$E(\text{no 1's}) = \sum_{i=1}^k (1 - \alpha_i)^l \simeq \sum_{i=1}^k e^{-l\alpha_i}$$

where  $k$  is the total number of columns.

Similarly, the expected number of columns with exactly 1 one is:

$$E(1 \text{ one}) \simeq \sum_{i=1}^k l\alpha_i e^{-l\alpha_i}$$



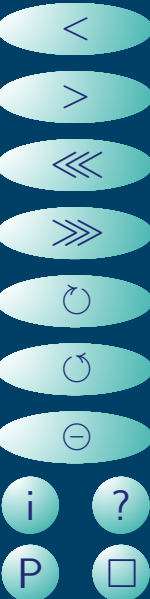
# Upper Bound

Now that we have found the expected number of columns with no ones, we still need to be able to come up with an upper bound for dependency. In fact, we need to determine if there exists some relationship between the dimension of the space, or factor base size, and the number of vectors, or  $B$ -smooth rows. If we let the number of vectors  $l = l(k)$  be a function of our dimension, then we can try to determine an upper bound for dependency.

Since we know that a trivial dependency occurs when the number of vectors and columns with no ones exceeds the dimension of the space, we have an ATD if:

$$l + \sum_{i=1}^k e^{-l\alpha_i} > k$$

How should  $l(k)$  grow for this inequality to hold?



# Some Heuristics

For the model in which we had  $\alpha_i = \frac{1}{p_i+1}$ , we showed this  $l \simeq \sqrt{k}$ . For various  $k$ , the table below shows values of  $l$  that satisfy the previously stated inequality.

| dimension $k$ | $l(k)$ | $\sqrt{k}$ | difference |
|---------------|--------|------------|------------|
| 100           | 15     | 10         | 5          |
| 1000          | 46     | 31.623     | 14.377     |
| 10000         | 124    | 100        | 24         |
| 100000        | 317    | 316.228    | .772       |



# Implementation

Currently, we have been able to implement a working sieve and are testing data to see if our heuristics are correct.

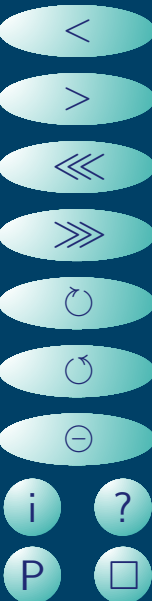
We factored the number 89612417583329 in this sieve and these are the results we got:

- Our factor base size was 2093
- Our smoothness bound was 38480
- $L = 1600$
- We found  $l = 429$  smooth numbers



# Implementation

After removing empty columns, removing solon columns and rows, and interating the process, we reduced a  $1600 \times 2093$  matrix to a  $65 \times 61$  matrix!





# References

1. Calkin, Neil. "Dependent Sets of Constant Weight Binary Vectors." *Combin. Probab. Comput.* 6, no. 3, 263-271, 1997.
2. Crandall, R. and C. Pomerance. *Prime Numbers A Computational Perspective*. Springer-Verlag, New York, 2001.

