

Monte Carlo Algorithms for Hardy-Weinberg

Proportions

Mark Huber,¹ Yuguo Chen,² Ian Dinwoodie,² Adrian Dobra² and Mike
Nicholas³

¹Department of Mathematics and ISDS, Duke University, Durham, NC 27708-0320,
USA

²Institute of Statistics and Decision Sciences, Duke University, Durham, NC
27708-0251, USA

³Department of Mathematics, Duke University, Durham, NC 27708-0320, USA

April 25, 2005

SUMMARY

The Hardy-Weinberg law is among the most important principles in the study of biological systems (Crow, 1988). Given its importance, many tests have been devised to determine if a finite population follows Hardy-Weinberg proportions (HWP). Because asymptotic tests can fail, Guo and Thompson (1992) developed an exact test, unfortunately, the Monte Carlo method they proposed to evaluate their test has a running

Key words: Hardy-Weinberg; Monte Carlo; Direct Sampling; Exact p value

time that grows linearly in the size of the population N . Here we propose a new algorithm whose expected running time is linear in the size of the table produced, and completely independent of N . In practice, this new algorithm can be considerably faster than the original method.

1. Hardy-Weinberg Proportions

The Hardy-Weinberg law is a cornerstone of population genetics, and it is very important to be able to test whether or not a given population conforms with this law (see for example Galbusera et al., 2000). Suppose that the population mates randomly and that there is no selection or mutation affecting the gene frequencies. Under these conditions, the Hardy-Weinberg law states that the frequency of genotypes will remain constant across generations, and gives a formula for these frequencies.

These are limiting frequencies that apply to infinite populations. For finite populations, the question of interest is whether or not the finite population is a random subset of a population that follows the Hardy-Weinberg law. Such a random subset has a distribution known as Hardy-Weinberg proportions (HWP). Testing if a finite population obeys HWP is an important first step in an analysis of the population. For example, Galbusera, et al. (2000) studied subpopulations of the endangered Taita thrush. Their first step was to determine if migration was occurring between the subpopulations by testing each to determine if the population was in HWP.

Paterson et al. (1998) examined various alleles in a population of Soay sheep in order to determine which alleles were selected for by the presence of a parasite. We

reanalyze part of their data in Example 2 below.

Exact p -values for testing if a population is in HWP goes back to Levene (1949). The idea is to look at a test statistic such as the probability of the allele frequencies observed in the data. Then the probability that a table drawn at random from HWP has a test statistic value lower than that of the data is an *exact p -value*.

Let N be the size of the finite population. The set of possible allele frequencies for the population is exponentially large in N . An exponential time network algorithm for determining this exact p -value deterministically exists (Aoki, 2003), but is only useful for values of N that are small (the example considered in the paper has $N = 30$).

For larger values of N , the only efficient means for testing if a finite population is drawn from HWP is Monte Carlo simulation. Guo and Thompson (1992) proposed two such Monte Carlo methods. The first was a direct Monte Carlo method that unfortunately only ran in pseudopolynomial time, which is discussed in Section 1.1. This slow running time prevented Guo and Thompson from using this method to analyze one of their examples (Example 1 considered below). The second approach was a Markov chain method, where the mixing time of the chain was guessed at but never fully determined. Improvements on their method concentrated on speeding up the Markov chain technique (Lazzeroni and Lange, 1997), but the direct Monte Carlo method remained in its original form.

In this paper we present a new Monte Carlo technique for this problem that is an improvement on the direct Monte Carlo method of Guo and Thompson. It is a direct method, not based on Markov chains, and is the first true linear time algorithm for

generating random variates exactly from the desired distribution in the sense that the time needed to run the algorithm is linear in the size of the table produced. In addition, this method is able to handle the constraint of one structural zero in the problem, which is discussed in Section 3.3. Furthermore, it is fast in practice as well as theory; we present later a relatively small data set where our method is seventeen times as fast as previous approaches.

To state the Hardy-Weinberg law precisely, consider a particular autosomal locus that is always one of m different alleles in the population. Suppose that allele A_i occurs with frequency p_i for i from 1 to m . Then the Hardy-Weinberg law states that allele combination A_iA_j (where $i < j$) occurs with frequency $2p_ip_j$ in the population, and A_iA_i occurs with frequency p_i^2 . These frequencies are limiting behavior for a large population. Now consider a finite population of N individuals and a particular autosomal locus. Since the locus is autosomal the genetics of the population can be described completely by writing out the $2N$ allele types found in the N members of the population. We will say that the population is in Hardy-Weinberg equilibrium or exhibits *Hardy-Weinberg proportions* (HWP) if this ordering is equally likely to be any permutation of the $2N$ alleles found in the population.

For our purposes, two individuals possessing the same allele combination at the locus of interest are indistinguishable from one another, and so we are actually only interested in the observed frequency of each pair of alleles.

Suppose that allele A_i occurs f_i times in the population. This is fixed, as any permutation of the alleles does not change the f_i . Let f_{ji} be the number of times

allele combination $A_j A_i$ ($i < j$) occurs. Then for all i and j from 1 to m :

$$f_i = 2f_{ii} + \sum_{j>i} f_{ji} + \sum_{j<i} f_{ij}. \quad (1)$$

Restricting ourselves to frequencies that meet the linear constraints in (1), the distribution of frequencies becomes (Levene, 1949):

$$\pi(f) = \frac{N!}{(2N)!} \left(\prod_{i \leq j} \frac{1}{f_{ji}!} \right) \left(\prod_{i=1}^m f_i! \right) 2^{\sum_{i<j} f_{ji}}. \quad (2)$$

Various tests exist to determine if a particular data set follows HWP. Large sample goodness of fit tests such as Pearson's χ^2 (Li, 1955) rely on asymptotic results that are not always valid for the data sets of the type we consider. Guo and Thompson (1992) proposed an exact p value test; evaluating their p value requires the ability to generate samples from the distribution π in (2). It is this problem of generating random variates in order to determine exact p values that will be our focus.

1.1 *Slow and fast direct generation of random variates*

Given that we derived π in (2) from the uniform permutation on $2N$ alleles, we can generate a random variate directly from π by first generating a random permutation of the $2N$ alleles and then just counting the frequencies f_{ij} directly (Guo and Thompson, 1992). We will refer to this as the *naive Monte Carlo* method for this problem.

Generation of the permutation of $2N$ elements requires $2N$ space and $2N$ draws of random uniforms. We are only interested in m choose 2 or $\Theta(m^2)$ different frequencies, but N might be exponentially large in the size of the problem instance. The problem requires $O(m \ln N)$ bits to write down the allele frequencies when N is encoded using binary notation for numbers, and so $O(m^2 \ln N)$ bits are required to write out the

final random table. Hence the input and output sizes are $O(m^2 \ln N)$. The running time of the Guo and Thompson algorithm is N , and so technically it is an exponential time algorithm since N can be exponentially large in $m^2 \ln N$.

In unary notation the integer a is written as a sequence of 1's of length a . For example the number 3 is 111 in unary. When the numbers in the input and output are encoded in unary the input size and output size becomes $O(m^2 N)$. When the running time of an algorithm is polynomial when the problem is encoded in unary, the algorithm is called *pseudopolynomial* (Wolsey and Nemhauser, 1999).

This pseudopolynomial running time for the original Monte Carlo algorithm has led various authors to consider Markov chains. Unfortunately, the Markov chain mixing time used in papers such as Guo and Thompson (1992) have mixing time $\Theta(N \ln N)$, and so are slower than directly drawing the permutation in the first place. Yuan and Bonney (2003) develop a better Markov chain, but do not fully analyze the mixing time. The Markov chain algorithms use $\Theta(m^2)$ space, compared to the $\Theta(N)$ space used by the naive Monte Carlo method.

In Section 3 we present two true linear time algorithms that only require $\Theta(m^2)$ time and space to generate a random variate from the desired distribution, with no dependence on N . We compare our algorithms to the naive Monte Carlo method and show that it is much faster in practice as well as theory.

2. The Problems

In this section we introduce several data sets that are typical for this field, along with the exact p value test of Guo and Thompson (1992).

Example 1. The following data set was extracted (Guo and Thompson, 1992) from Rhesus data in Cavalli-Sforze and Bodmer's *The Genetics of Human Population* (1971).

[Figure 1 about here.]

For this table f , $\pi(f) = 10^{-54.7}$. The exact p value of Guo and Thompson (1992) works by computing the probability that a random table chosen from π has probability at most $\pi(f)$. More precisely, let $\mathbf{F} = (f_1, \dots, f_m)$ and let

$$S_{\mathbf{F}} = \{g = (g_{ij}) : 2g_{ii} + \sum_{j>i} g_{ji} + \sum_{j<i} g_{ij} = f_i \quad \forall i\}. \quad (3)$$

Then the exact p value is just

$$p = \sum_{g \in S_{\mathbf{F}} : \pi(g) \leq \pi(f)} \pi(g). \quad (4)$$

Guo and Thompson analyzed this data using both χ^2 and their exact p value technique. The exact p value was approximated by Monte Carlo methods, hence the speed of this analysis is directly proportional to the speed at which random variates can be generated.

Example 2. In Paterson et al. (1998), a population of Soay sheep is studied to determine how resistance to a particular parasite affected sheep survival. The resistance

to parasite levels was believed to arise from an interaction of three of the five loci studied. The first step in such a determination was to show that individually, five genetic loci of interest were in Hardy-Weinberg equilibrium. This step was necessary to show that none of the loci of interest were affecting resistance to the parasite on their own; the effect had to arise from linkages between the genes.

We did not have access to the complete initial data, instead what was reported in Paterson et al. (1998) were the genetic frequencies for a particular genetic marker, OLADRB.

[Figure 2 about here.]

Our goal for this example was not to reanalyze the data of Paterson et al. (1998), but rather to examine how quickly our method worked on this particular example versus the Guo and Thompson method they used in their analysis.

Example 3. This genotype data for Gaucher disease comes from le Coutre et al. (1997). In Beutler et al. (1995), it is stated that the genotype pair IVS2+1/IVS2+1 is lethal, and therefore constitutes a structural zero in the triangular table of genotypes (structural zeros are discussed further in Section 3.3). The genotype data is as follows:

[Figure 3 about here.]

where the – indicates the presence of a structural zero.

3. Fast Generation of Random Variates

We now present an algorithm for sampling from Hardy-Weinberg proportions using $m(m+1)/2$ draws from various hypergeometric distributions. A hypergeometric draw can be accomplished using 4.2 (in expectation) uniform random variables, making this algorithm much faster than previous methods (Hörmann, 1994).

Guo and Thompson (1992) performed a Monte Carlo study on the Rhesus example of the previous section, where $N = 8297$, and $m = 7$, using generation of the entire permutation. This method requires 8192 draws of uniform random variables to generate one permutation which gives one table of frequencies, whereas our method takes only 28 hypergeometric draws (or roughly 75 uniform draws) for a complete set of frequencies, a reduction in the number of uniforms generated by a factor of 100.

Consider writing the frequencies in a lower triangular matrix. Our method fills in the matrix one column at a time from left to right. For a given column, we first tackle a diagonal entry such as f_{11} . This will require two draws of a hypergeometric. Then we fill in each element in the column below, each with a single hypergeometric. Once a column is filled, the numbers for the remaining alleles can be updated accordingly. The process then repeats until we have entered figures in all columns.

3.1 *Hypergeometric random variables*

Several acceptance/rejection methods exist for generating from the hypergeometric distribution in constant time. For instance, Kachitvichyanukul and Schmeiser (1985) use a simple envelope function, Stadlober (1990) uses a ratio of uniforms and Stadlober and Zechner (1999) utilize patchwork rejection. More generally, the hy-

pergeometric distribution is log-concave, that is, for X a hypergeometric random variable, $P(X = k)^2 \geq P(X = k - 1)P(X = k + 1)$ for all k . This means that constant time results in Devroye (1987) and Hörmann (1994) for generating variates from discrete log-concave distributions can be used.

In particular, Theorem 2 of Hörmann (1994) indicates that at most 4.2 uniform variates are needed (on average) for a single hypergeometric draw.

A hypergeometric random variable can be viewed in the following fashion. Consider t balls dropped uniformly at random into $s = r + g$ slots (each slot holding at most 1 ball) where r of the slots are colored red, and g of the slots are colored green. Then if X is the number of balls that fall in red slots, then we say that X has a hypergeometric distribution with parameters s, r , and t . We write $X \sim HG(s, r, t)$.

This view of hypergeometric random variable relates to the HWP problem as follows. Suppose that the alleles of type 1 are our balls. The slots are the $2N$ possible places that the alleles can be placed. Each of the N members of the population is assigned two slots corresponding to the two alleles for that member. Color the set of first slots assigned to each person red, and the set of second slots assigned to each person green.

Now for a person to have allele type A_1A_1 , both the red and green slots assigned to that person must be occupied by balls. Consider a two step procedure. First, randomly decide how many of the allele type 1 balls fall in red slots. This number X will have a hypergeometric distribution, say $X \sim HG(2N, N, f_1)$. Now give second slots where the first slots are occupied by a ball the color yellow. The remaining

second slots are colored blue. There are X yellow slots, and $N - X$ blue slots. Dump the remaining $f_1 - X$ balls into these slots. Now the number of members of the population (call it Y) that have received A_1A_1 allele type is again hypergeometric, so $Y|X \sim HG(N, X, f_1 - X)$. This number Y is the entry f_{11} in the table. This is illustrated in Figure 4, where $N = 6$, $f_1 = 6$, $X = 3$, and $Y = 2$.

[Figure 4 about here.]

Now given f_{11} , consider the number of members of the population that receive A_1A_2 alleles. The number of allele 1 balls that only occupy one of a member's two slots will be $f_1 - 2f_{11}$. Color empty slots that are next to such an allele 1 ball yellow. Now f_2 allele 2 balls are dropped randomly into the $2N - f_1$ slots that remain, of which $f_1 - 2f_{11}$ are colored yellow. If an allele 2 ball lands in a yellow slot, that member has genotype A_1A_2 . Hence the number of such people has a hypergeometric distribution, $HG(2N - f_1, f_1 - 2f_{11}, f_2)$.

In the same fashion, $f_{13}, f_{14}, \dots, f_{1m}$ can be chosen until the contents of the entire column are filled in. This idea is the heart of an induction that makes the above argument formal, and allows the completion of a column.

3.2 *Column by column*

Once the first column is filled in, the next column can be tackled in a similar fashion, and the algorithm proceeds by filling in the columns of the table one at a time. We now verify that this procedure is valid. Suppose that we are trying to generate a random permutation of the alleles conditioned on the first column of

frequencies being fixed. That is, the counts f_{11} through f_{1m} are known.

Such a permutation can be constructed as follows. For each member with allele A_1A_j , $j \neq 1$, randomly permute the genotype so that it is either A_1A_j or A_jA_1 . Now take each of the f_{1j} members with genotype A_1A_j or A_jA_1 and randomly place them among the N members of the population. Finally, for the remaining alleles, randomly place them among the $2(N - f_{11} - \dots - f_{1m})$ slots that remain.

Any permutation satisfying f_{11} through f_{1m} can be constructed in this fashion, and each of these permutations will have probability

$$\frac{1}{\prod_{j=2}^m 2^{f_{1j}}} \cdot \frac{N!}{\prod_{j=1}^m f_{1j}!} \cdot \frac{1}{\{2(N - f_{11} - \dots - f_{1m})!\}} \quad (5)$$

of occurring. This is constant when conditioned on f_{11} through f_{1m} , and so this is a valid procedure for generating permutations uniformly at random conditioned on the values in the first column. Again to formally complete the argument an induction on the column number is necessary.

The procedure is: first fill in the first column of the matrix. Then update the numbers of alleles f_2 through f_m by subtracting off f_{12} through f_{1m} respectively. Finally, update N by subtracting $f_{11} + \dots + f_{1m}$. The result is a new problem that can be solved recursively. Since the base case where we have no columns is easy to sample from, it is an easy proof via induction that this procedure generates directly from π .

Generating from Hardy-Weinberg Proportions

Input: N, f_1, \dots, f_m

1. **For** i from 1 to m
2. **Choose** $a_i \leftarrow HG(2N, N, f_i), f_{ii} \leftarrow HG(N, a_i, f_i - a_i)$
3. **Let** $N \leftarrow N - (f_i - f_{ii}), f_i \leftarrow f_i - 2f_{ii}, b \leftarrow 2N - f_i$
4. **For** j from $i + 1$ to m
5. **Choose** $f_{ji} \leftarrow HG(b, f_i, f_j)$
6. **Let** $b \leftarrow b - f_j, f_j \leftarrow f_j - f_{ij}, f_i \leftarrow f_i - f_{ij}$

3.3 Structural zeros

The structure of the data can contain additional constraints beyond (1). For instance, a particular allele type A_iA_i might be lethal to the organism born with it. Example 3 is an example of this type of data set.

Such a lethal combination implies that the data f_{ii} is forced to be 0. This constraint is referred to as a *structural zero* of the problem. Selection is occurring here, therefore the population does not satisfy Hardy Weinberg, and cannot be expected to follow HWP. Instead, it is reasonable in this case to test whether the population follows HWP conditioned on the f_{ii} entry being zero. Such a conditional distribution exists as long as $\sum_{j \neq i} f_j > f_i$.

The gene labels can be permuted so that f_{11} is the entry constrained to be 0. To sample from permutations where each of the A_1 alleles reside in different members of the population, we simply begin the algorithm with $f_{11} = 0$ for the choice of first diagonal entry and proceed as before. Since the algorithm proceeds by filling in the

entries one at a time conditioned on the entries previously chosen, this procedure automatically returns a table drawn from the correct conditional distribution.

3.4 *Running Time*

The number of hypergeometric draws employed by our method is m choose 2 plus m , or $(m^2 + m)/2$. Given that we can take a hypergeometric draw on average using 4.2 uniform draws (Hörmann, 1994), we immediately have the following.

THEOREM 1. *An upper bound on the expected number of uniforms used by our method is $2.1(m^2 + m)$.*

4. Empirical Results

Guo and Thompson (1992) were unable to use their exact method on Example 1 because it took too much time. Instead they used a Markov chain method where they did not fully analyze the mixing time. Advances in computing speed allow us to run Guo and Thompson's original method alongside our own and compare their running times. Also, we created artificial data sets by multiplying the data in each of our examples by a factor of 100. These artificial data sets illustrates nicely how the Guo and Thompson method running time depends on data size, while ours does not.

The running times for p -values here are the times needed to take 10^6 samples (except for the Example 1 times 100 data set, where the time is an estimate of how long 10^6 samples would take based on the first 1000 samples). Note that the exact p values found by each method are the same: both methods give the same answer, what is different is just how quickly the estimate for the p value is generated. The

confidence intervals for the p -values are presented at the 99.9% confidence level.

For Example 1, the number of uniforms needed to be generated was significantly less, however, there is fixed overhead in calling subroutines that made the speedup about 17, less than expected. With the artificial data set where each entry is multiplied by 100, the speedup rises to 1400.

In Example 2, the speedup is about 2 for the original problem, and for Example 3, our method actually takes longer, because N is so small. However, the exact method of Aoki (2003) was successfully used on a problem where $N = 30$, and so Monte Carlo methods need not be used at all when N is this small. Of course, once each data set is multiplied by 100, our method again becomes faster, showing speedups of 97 and 6 respectively.

[Figure 5 about here.]

5. Conclusions

Both theoretically and in practice, the new algorithm of Section 3 is the fastest method at present for generating draws exactly from Hardy-Weinberg proportions, except when the size of the population is very small, in which case Monte Carlo methods need not be used at all. The fact that the running time only depends on the number of alleles and is independent of the total population is a sharp improvement over previous methods whose running time grew linearly in the total population.

ACKNOWLEDGEMENTS

Research reported was partially supported by the National Science Foundation through SAMSI under DMS-0112069, by NSF Grant DMS-0200888, and NSF Grant DMS-0203762. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

- Aoki, S. (2003). Network algorithm for the exact test of Hardy-Weinberg Proportion for multiple alleles. *Biometrical Journal* **45**, 471–490.
- Beutler, E., Gelbart, T., Demina, A., Zimran, A. and le Coutre, P. (1995). Five new Gaucher disease mutations. *Blood Cells, Molecules, and Diseases* **21**, 2–24.
- Crow, J. E. (1988). Eighty years ago: The beginnings of population genetics. *Genetics* **119**, 473–476.
- Devroye, L. (1987). A simple generator for discrete log-concave distributions. *Computing* **39**, 87–91.
- Galbusera, P., Lens, L., Schenck, T., Waiyaki, E. and Matthysen, E. (2000). Genetic variability and gene flow in the globally, critically-endangered Taita thrush. *Conservation Genetics* **1**, 45–55.
- Guo, S. and Thompson, E. A. (1992). Performing the exact test of Hardy-Weinberg proportion for multiple alleles. *Biometrics* **48**, 361–372.
- Hörmann, W. (1994). A universal generator for discrete log-concave distributions.

Computing **52**, 89–96.

Kachitvichyanukul, V. and Schmeiser, B. (1985). Computer generation of hypergeometric random variates. *J. of Stat. Comp. and Sim.* **22**, 127–145.

Lazzeroni, L. C. and Lange, K. (1997). Markov chains for Monte Carlo tests of genetic equilibrium in multidimensional contingency tables. *Ann. Stat.* **25**, 138–168.

le Coutre, P., Demina, A., Beutler, E., Beck, M. and Petrides, P. (1997). Molecular analysis of Gaucher disease: distribution of eight mutations and the complete gene deletion in 27 patients from Germany. *Hum. Genet.* **99**, 816–821.

Levene, H. (1949). On a matching problem arising in genetics. *Annals of Mathematical Statistics* **20**, 91–94.

Stadlober, E. (1990). The ratio of uniforms approach for generating discrete random variates. *J. Comput. Appl. Math.* **31**, 181–189.

Stadlober, E. and Zechner, H. (1999). The patchwork rejection method for sampling from unimodal distributions. *ACM Trans. on Modeling and Comp. Sim.* **9**, 59–80.

Wolsey, L. A. and Nemhauser, G. L. (1999). *Integer and Combinatorial Optimization*. Wiley-Interscience.

Yuan, A. and Bonney, G. (2003). Exact test of Hardy-Weinberg equilibrium by Markov chain Monte Carlo. *Mathematical Medicine and Biology* **20**, 327–340.

List of Figures

1	Genotype data at Rhesus locus (Cavalli-Sforza and Bodmer, 1971) . .	19
2	OLADRB gene in Soay sheep	20
3	Gaucher disease data (le Coutre et al., 1997)	21
4	Choosing f_{11}	22
5	Empirical running times	23

1236								
120	3							
18	0	0						
982	55	7	249					
32	1	0	12	0				
2582	132	20	1162	29	1312			
6	0	0	4	0	4	0		
2	0	0	0	0	0	0	0	
115	5	2	53	1	149	0	0	4

Figure 1. Genotype data at Rhesus locus (Cavalli-Sforza and Bodmer, 1971)

Allele	205	213	257	265	267	269	276	287
Observed	506	271	566	319	378	3	337	18

Figure 2. OLADRB gene in Soay sheep

0						
5	2					
2	0	0				
1	0	0	—			
0	0	0	0	0		
0	1	0	0	0	0	
10	2	0	0	1	0	1

Figure 3. Gaucher disease data (le Coutre et al., 1997)

member	1	2	3	4	5
color	<input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/>
	R G	R G	R G	R G	R G

Drop allele 1 balls in red or green, count number in red slots

color	<input type="checkbox"/> <input type="checkbox"/>	<input checked="" type="checkbox"/> <input type="checkbox"/>	<input checked="" type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/>	<input checked="" type="checkbox"/> <input type="checkbox"/>
	R B	R Y	R Y	R B	R Y

Drop remaining allele 1 balls in yellow or blue slots

color	<input type="checkbox"/> <input type="checkbox"/>	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
	R B	R Y	R Y	R B	R Y

Figure 4. Choosing f_{11}

Problem	Running Time (seconds)		Exact <i>p</i> -value
	GT (1992)	Our method	
Example 1	5732	332	$.714 \pm 0.001$
Example 1 times 100	$1.1 \cdot 10^6$	781	-
Example 2	811	239	$.00779 \pm .00001$
Example 2 times 100	74700	274	-
Example 3	22	368	$.0412 \pm 0.0002$
Example 3 times 100	1750	293	-

Figure 5. Empirical running times