Journal of Modern Mathmematics and Statistics 1 (1-4): 15-23, 2007
ISSN: 1994-5388


# On the Philosophy of Statistical Bounds: A Case Study on a Determinant Algorithm

[1]Soubhik Chakraborty, [2]Charu Wahi, [3]Suman Kumar Sourabh,
[4]Lopamudra Ray Saraswatid and [1]Avi Mukherjee
[1]Department of Applied Mathematics,
[2]Department of Computer Science, BIT Mesra, Ranchi-835215, India
[3]University Department of Statistics and Computer Applications,
T.M. Bhagalpur University, Bhagalpur-812007, India
[4]International Institute of Population Sciences, Mumbai-400088, India

**Abstract:** Under the umbrella of statistical algorithmic complexity (which some authors call stochastic arithmetic) , it makes sense to talk about statistical bounds (asymptotic) and their empirical estimates over a finite range (a computer experiment cannot be run for infinite input size!), the so called empirical O, which were informally introduced in Chakraborty and Sourabh where it was shown that they make average complexity more meaningful. The present study shows that these concepts can be used effectively in worst cases as well as in best cases besides average cases with a case study on an efficient determinant algorithm.

**Key words:** Determinant, statistical algorithmic complexity, statistical bound, empirical O

## INTRODUCTION

Under the umbrella of statistical algorithmic complexity (which some authors call stochastic arithmetic (Vignes, 1993), it makes sense to talk about statistical bounds (asymptotic) and their empirical estimates over a finite range (a computer experiment cannot be run for infinite input size!), the so called empirical O. These concepts were informally introduced in Chakraborty and Sourabh (2007), where it was shown that they make average complexity more meaningful. The present study shows that these concepts can be used effectively in worst cases as well as in best cases besides average cases with a case study on an efficient determinant algorithm. It is shown here that worst case complexity is basically a safe and conservative science which is misinterpreted as useful just because of the *guarantee* element which can be provided without being too conservative (nature is not an antagonist!). Similarly, it also guards against making tall unrealistic claims in best cases and prevents ending up in a false position. And of course it is again found ideal for the average case.

There are two serious objections to applying mathematical bounds in average complexity analysis. In mathematical bounds operations are counted and a separate bound is provided to a specific operation type. This means we must know which operation is pivotal before applying expectation to it for average case analysis. The objections are:

- In complex codes (e.g. in partial differential equations) how do you know which operation is pivotal?
- To apply expectation some probability distribution (generally uniform) is assumed.

How realistic is the probability distributional assumption over the problem domain?

The ideal strategy is to *weigh* rather than *count* the operations, take *all of them collectively* into a conceptual statistical bound and then set out to *estimate* this non-trivial realistic bound by physically conducting a computer experiment in which the response is a complexity suitably chosen to represent the weights in some sense (such as time). The better the design and analysis of this special kind of our computer experiment, the greater is the credibility of the bound-estimate, the so called "empirical-O". The link between algorithmic complexity and computer experiments (Chakraboorty and Sourath, 2007; Chakraboorty *et al*., 2007). It should be kept in mind that average itself is a statistical term and that the most popular and useful average (arithmetic mean) is itself only a special case of *weighted* mean where

**Corresponding Author:** Soubhik Chakraborty, Department of Applied Mathematics, BIT Mesra, Ranchi-835215, India

the *weights* are frequencies. Although ideal for average case analysis, the present paper utilizes these concepts for worst case and best case as well profitably confirming their robustness in a realistic way over the problem domain. Our study is conducted in three phases.

**Phase 1:** Here we have used the concept of an empirical O for worst case analysis by selecting the determinant elements as U[0, 1] variates so that the value of the determinant is always non-zero as the probability for positive and negative real numbers to cancel exactly is zero during the evaluation of the determinant.

**Phase 2:** Here the empirical O explains average case analysis where the determinant elements are selected as discrete uniform variates in the range [0, 1, 2, ….n-1] so that there is a positive probability for the value of the determinant to be zero as it is quite possible for positive and negative integers to cancel exactly during the evaluation of the determinant.

**Phase 3:** This covers best case analysis in which first the determinant elements are selected as discrete uniform variates in the range [0, 1, 2, ….n-1] and then all elements of a randomly selected column are made zero so that the value of the determinant is always zero.

We close this section by formally defining statistical bounds and their empirical estimates, the so called "empirical O".

**Statistical bound (definition):** Let $w_{ij}$ is the weight of operation of type i in the j-th repetition and y is a "stochastic realization" (Chakraboorty and Sourath, 2007; Chakraboorty *et al*., 2007)) of the deterministic $T = \Sigma 1. w_{ij}$ where, we count one for each operation repetition irrespective of the type, the statistical lub of the algorithm is the asymptotic lub on y expressed as a function of n, the input parameter characterizing the input size.

**Empirical O (definition):** We define an empirical O as "the O that corresponds to the simplest model fitted to (time) complexity data that is neither an underfit nor an overfit." More specifically, it corresponds to a model that does not invite the serious problem of ill-conditioning nor does it lead to a loss of predictive power.

**A few grains of salt:**

- From a philosophical point of view, a statistical bound differs from the empirical O in as much as God differs from His idol, the latter being only an "estimate" or an imitation of The Truth over a finite range. And just as a skilled architect builds a beautiful idol inside a nice temple to assist peaceful worshiping, so does the statistician who conceives a non-trivial and conceptual statistical bound and sets out to estimate it through empirical O over a finite range (the computer experiment cannot be run for infinite input) and thereby concentrate on increasing the credibility of the estimate through a proper design and analysis of this special kind of computer experiment of his whose response is a complexity such as time. Moreover, we are not the only authors preaching such a "statistical algorithmic complexity". There are others also. See for example J. Vignes, A Stochastic Arithmetic for Reliable Scientific Computation, Mathematics and Computers in Simulation, 1993, The reader is strongly encouraged to make a google search for terms like "stochastic arithmetic", "arithmetic quality", "numeric quality" to find out more. However there is a novelty in our approach especially the way we have linked it with computer experiments.

- Except for the fact that both theoretical and empirical O have the leading functional term inside the O, there is no other similarity. We hope the reader will appreciate the distinction. Theoretical O is a mathematical bound. Empirical O is a "bound-estimate" and not itself a bound. It is estimating the (asymptotic) non-trivial and conceptual statistical bound based on operation weights over a finite range [a, b] which becomes a feasible range if it catches a large number of live data sets within its range. As mentioned by Chakraboorty and Sourath (2007) empirical O cannot be a bound because: All bounds are asymptotic while it holds only over a finite range and more importantly because, it emerges from an empirical model which is affected by the statistician's subjective personal opinion and a bound cannot be subjective. Hence, the comparison should be meaningful if it is between mathematical and statistical bounds (Table 1D).
- The term "simplest model" used in the definition of empirical O needs to be clarified in such a way that empirical O looks like a bound estimate. An empirical model is simpler than another if the O of the corresponding theoretical model is stronger in the scale of algorithmic complexity comparison. (i.e. $\ldots O(n) < \ldots O(n\log n) \ldots O(n^2) \ldots$) Chakraboorty and Sourath (2007). Thus, if M is an empirical model selected, the statistician asks: What would have been the O if M were a theoretical statement and the range extended to $\infty$? If it is O(g(n)) we agree to call it

empirical O and rightfully attach a subscript emp to put $y = O_{emp}(g(n))$ remembering once again that it is neither a bound nor does it estimate a mathematical bound in any sense whatsoever. It is clearly a "bound-estimate" and from the earlier discussion it is clear that a non-trivial statistical bound is being cleverly guessed out in that when a program is run, the operations are collectively implemented which is what it should be in a statistical bound. In the statistical bound the weights are conceptual while in the empirical estimate the weights are assigned some numerical values. The statistician looks for "system invariance" and reports whether he/she gets it or otherwise in the algorithm in question.

## MATERIALS AND METHODS

Think of an algorithm for evaluating a determinant which has the provision for abrupt termination if at any stage it is detected that the value of the determinant is zero. It can also handle small elements efficiently (see the "else.." statement in step 3).

Let $a_{ij}$ be the element in the i-th row and j-th column of an nxn square matrix A.
To get det(A) follow the following steps:-

**Step 1:** Set j=1 and c=0
**Step 2:** Examine Column j
If $Max|a_{ij}| = 0$ where $i \geq j$, declare that det(A)=0 and stop. Else go to step-3
**Step 3:** Set $P= Max|a_{ij}|$, $i \geq j$. Is i=j? If yes, go to step-4 else interchange row i with row j.
In case of an interchange set c=c+1.
**Step 4:** Using elementary row transformations make the elements below the diagonal element in column j as zeroes.
**Step 5:** j=j+1. If $j \leq n$ goto step-2 else go to step-6.
**Step 6:** Deliver det(A) as the product of the diagonal elements of the triangular final form of A multiplied by $(-1)^c$ and stop.

**Worst Case Complexity of the Determinant algorithm: What makes a statistical bound realistic?**

**Lemma:** The theoretical worst case complexity of the determinant algorithm is $O(n^4)$.

**Proof:** Refer to the BASIC code in Section IIB depicting the algorithm clearly.

The most important statement is the line number 160. This contains an IF…THEN…ELSE statement. The time of an IF…THEN…ELSE statement is the time to evaluate the condition plus maximum of the two times, when the condition is true or when the condition is false. Time to evaluate the condition is generally O(1) (Aho *et al*., 2000).

Now, in our case, maximum time will be taken when the condition is false, which corresponds to line number 245. So, let us concentrate on line number 245. There is a FOR loop over I which executes n times; inside it again there is another IF…THEN…ELSE statement. By previous argument, this should take $O(n^2)$ worst case time considering the nested loop over L and M through the subroutine (line number 400 to 460).

Hence the complexity considering the I loop, which executes n times, is $O(n^3)$. Now, we come back to line number 160 and observe that this whole thing is happening inside the J loop, which in the worst case will execute n times. This leads to an $O(n^4)$ complexity.
We further make the following observations:-

- Lines numbered 120 to 140 corresponds to O(n) and $O(n)+O(n^3) = O(n^3)$, by the sum rule of big O. This is therefore the worst case complexity inside the J loop.
- Outside the J loop, we again get an O(n) complexity for line numbers 190 to 210 (loop over B).
  So, finally, we have $O(n^4)+ O(n)=O(n^4)$, by the sum rule of big O.
- All assignment statements such as M=0 consumes O(1) time and have not been taken in the analysis as they would not affect the big O due to the sum rule.
- Operations inside a loop are regarded as a "compound" operation, whatever be the time.

In conclusion, the worst case complexity of our algorithm is $O(n^4)$, completing the proof.

**Phase 1: A Contradiction inWorst case experimental analysis!!!:** Here is a GWBASIC (version 2.02) program which we executed in Microsoft Windows XP Professional 2002 operating system on Intel® Pentium4 with CPU 1.50 GHz and 256 MB of RAM.

```
5    REM PROGRAM FOR EVALUATING A DETERMINANT:CLS
10   INPUT "enter n";N
20   DIM C(N,N):RANDOMIZE TIMER
30   FOR I=1 TO N
40   FOR J=1 TO N
50   C(I,J)=RND
70   NEXT J
80   NEXT I
85   ST=TIMER
90   J=1:CONS=0
100  M=0
120  FOR I=1 TO N
130  IF I>=J AND ABS(C(I,J))>=M THEN M=ABS(C(I,J))
140  NEXT I
150  IF M=0 THEN DET=0:GOTO 225
```

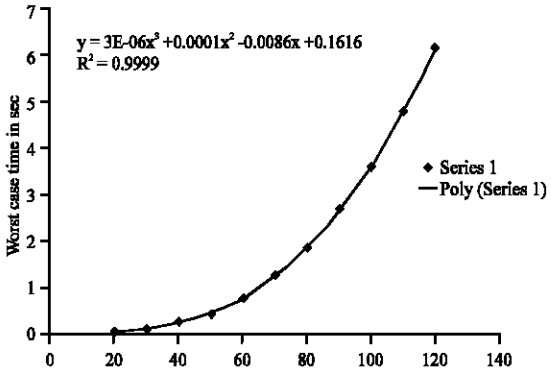


Fig. 1: Worst case time in sec versus n

```
160  IF M=ABS(C(J,J)) THEN GOSUB 400 ELSE 245
170  J=J+1:IF J<=N THEN 100
180  D=1
190  FOR B=1 TO N
200  D=D*C(B,B)
210  NEXT B
220  DET=(-1)^CONS*D
225  ET=TIMER
230  PRINT "det=";DET
235  PRINT "run time in sec=";ET-ST
240  END
245  FOR I=1 TO N
250  IF I>J AND M=ABS(C(I,J)) THEN 260 ELSE 310
260  CONS=CONS+1
270  FOR T=1 TO N
280  SWAP C(J,T),C(I,T)
290  NEXT T
300  GOSUB 400
310  NEXT I
320  GOTO 170
400  REM subroutine for triangularization
410  FOR L=1 TO N
420  IF L<=J THEN 460
425  G=C(L,J)
430  FOR M=1 TO N
440  C(L,M)=C(L,M)-G*C(J,M)/C(J,J)
450  NEXT M
460  NEXT L
470  RETURN
```

Worst case experimental results for determinant elements from U[0,1] distribution with non zero value of the determinant (based on GWBASIC interpreter version 2.02; run in Pentium4 on MS-Windows XP 2002) (Fig. 1):-

Though the cubic fit is excellent, recall that empirical O is a bound-estimate and all bounds are asymptotic; hence we cross verify the result for higher n in a different system environment which would also establish the "system invariance" of statistical bounds before putting anything concrete.

## CHANGING THE SYSTEM ENVIRONMENT

| System specifications | |
|---|---|
| Processor | HP Pentium ® 4 CPU 3.00 GHz |
| Hard Disk | 80 GB |
| RAM | 448 MB |
| Operating System | Windows XP Professional |

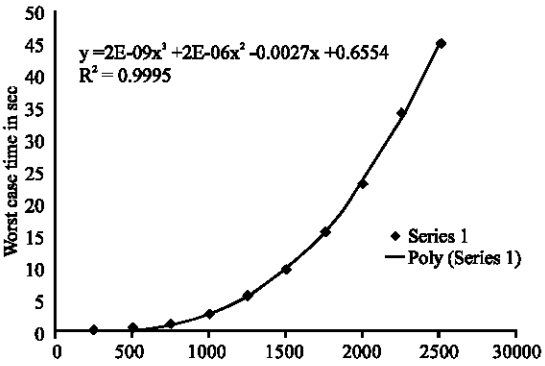


Fig. 2: Worst case time in sec versus n

Version 2002
Service Pack 2
Remark. The C++ used is given in Appendix 3:

Table 1 shows the Worst Case Complexity in Computing the Determinant of order n in the new system. All times are in seconds.

**Interpretation:** Evidently $T(n)=O_{emp}(n^3)$ for the worst case from the definition of empirical O. The BASIC code showed the results to be true for small n while we have just verified it for large n. Hence the result is true over a feasible range. No Kussmaul protection* (D-optimality) against a polynomial of degree four is necessary. Also system invariance of the statistical bound is established at least in the present algorithm (Fig. 2).

**Explanation and utility of the contradiction:** The contradiction between theory and practice occurs because of a "loss of dominance" of the computing operations and our inclination to weigh rather than count them which is justified. Another reason could be that we took all loops to execute to their maximum which is quite a conservative approach. The algorithm certainly performs better than we thought! As a straight application, one can think of a product of determinants each of order n with the number of determinants in the product being equal to n or its square or even higher. Use of empirical O in this case can prevent ill-conditioning, if any. A system of simultaneous equations AZ = C is said to be ill conditioned if small errors in the elements of A and/or C lead to large fluctuations in the solution vector Z. If we take $A=X^TX$ and $C=X^TY$ then Z is the solution of the familiar least square equations. Do you now see the seriousness of the problem in a computer experiment? For a helpful discussion on the general problem of ill-conditioning with polynomial regression as a special case, (Sebev, 1987) which also gives a review of optimal

Table 1: Worst Case Complexity in Computing the Determinant of order n in the new system

| n | Trial 1 | Trial 2 | Trial 3 | Trial 4 | Trial 5 | Trial 6 | Trial 7 | Trial 8 | Trial 9 | Trial 10 | Mean Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 250 | 0.031 | 0.032 | 0.047 | 0.032 | 0.031 | 0.031 | 0.031 | 0.031 | 0.047 | 0.031 | 0.0344 |
| 500 | 0.266 | 0.282 | 0.282 | 0.266 | 0.281 | 0.282 | 0.266 | 0.281 | 0.281 | 0.281 | 0.2768 |
| 750 | 0.984 | 0.969 | 0.968 | 0.953 | 0.969 | 0.985 | 0.953 | 0.938 | 0.985 | 0.938 | 0.9642 |
| 1000 | 2.578 | 2.516 | 2.531 | 2.469 | 2.469 | 2.5 | 2.484 | 2.516 | 2.484 | 2.453 | 2.5 |
| 1250 | 5.344 | 5.36 | 5.297 | 5.312 | 5.312 | 5.328 | 5.281 | 5.312 | 5.313 | 5.296 | 5.3155 |
| 1500 | 9.547 | 9.531 | 9.546 | 9.562 | 9.515 | 9.531 | 9.61 | 9.531 | 9.484 | 9.516 | 9.5373 |
| 1750 | 15.313 | 15.313 | 15.328 | 15.297 | 15.282 | 15.375 | 15.328 | 15.343 | 15.407 | 15.359 | 15.3345 |
| 2000 | 22.813 | 22.812 | 22.937 | 22.859 | 22.906 | 22.922 | 23 | 22.922 | 22.937 | 22.922 | 22.903 |
| 2250 | 32.61 | 32.719 | 32.672 | 32.61 | 32.75 | 33.078 | 38.109 | 40 | 32.719 | 32.656 | 33.9923 |
| 2500 | 44.766 | 44.766 | 44.656 | 46.953 | 44.781 | 44.719 | 44.688 | 44.781 | 44.766 | 44.812 | 44.9688 |

Table 2: Average Case Complexity in Computing the Determinant of order n. All times are in seconds

| n | Trial 1 | Trial 2 | Trial 3 | Trial 4 | Trial 5 | Trial 6 | Trial 7 | Trial 8 | Trial 9 | Trial 10 | Mean Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 250 | 0.047 | 0.031 | 0.031 | 0.047 | 0.031 | 0.047 | 0.031 | 0.047 | 0.031 | 0.031 | 0.0374 |
| 500 | 0.281 | 0.266 | 0.281 | 0.313 | 0.265 | 0.266 | 0.266 | 0.265 | 0.265 | 0.266 | 0.2734 |
| 750 | 0.922 | 0.922 | 0.937 | 0.922 | 0.906 | 0.968 | 0.922 | 0.922 | 0.906 | 0.938 | 0.9265 |
| 1000 | 2.578 | 2.485 | 2.437 | 2.5 | 2.453 | 2.516 | 2.469 | 2.485 | 2.453 | 2.438 | 2.4814 |
| 1250 | 5.297 | 5.203 | 5.25 | 5.266 | 5.234 | 5.219 | 5.265 | 5.281 | 5.281 | 5.265 | 5.2561 |
| 1500 | 9.359 | 9.437 | 9.437 | 9.563 | 9.516 | 9.406 | 9.422 | 9.5 | 9.407 | 9.453 | 9.45 |
| 1750 | 15.313 | 15.375 | 15.141 | 15.25 | 15.094 | 15.125 | 15.188 | 15.141 | 15.203 | 15.156 | 15.1986 |
| 2000 | 22.64 | 22.656 | 22.641 | 22.687 | 22.718 | 22.735 | 22.656 | 22.625 | 22.718 | 22.656 | 22.6732 |
| 2250 | 32.282 | 32.344 | 32.313 | 32.219 | 32.375 | 32.343 | 32.328 | 32.281 | 32.235 | 32.375 | 32.3095 |
| 2500 | 44.094 | 44.204 | 49.234 | 44.093 | 44.125 | 44.032 | 44.125 | 44.266 | 44.531 | 44.094 | 44.6798 |

*see an interesting example of Kussmaul protection in S. Chakraborty and K. K. Sundararajan, InterStat,March2006#5 (http://interstat.statjournals.net). We thank Dr. Debasis Kundu (IIT, Kanpur) for the editorial handling for InterStat

designs especially D-optimality. Appendix 2. For polynomial fits, ill-conditioning is severe when the degree of the polynomial is at about six or higher. Therefore it is certainly not wise to fit a polynomial of a very high degree unless there is a corresponding enhancement in the predictive power. Hence, as a second but perhaps more important application, if we are trying to approximate a complex curve by a polynomial (which we can by Weierstrass theorem over a continuous range; time is continuous!) for an arbitrary algorithm's complexity estimation case, it makes sense to select the simplest polynomial that fits the data well and design the computer experiment optimally so that the loss of predictive power is tolerable. If p be the polynomial's degree, which may have to be Kussmaul-protected* against a polynomial of degree p+1 or p+2 say, where p<6, we would joyfully put $T(n) = O_{emp}(n^p)$. Our point is proved.

**Phase 2: Average Case Analysis Through Statistical bounds:** From what we have said earlier, we shall drop traditional analysis and explain average case with empirical O (Table 2).

System Specifications
Processor HP Pentium ® 4 CPU 3.00 GHz
Hard Disk 80 GB
RAM 448 MB
Operating System Windows XP Professional
Version 2002
Service Pack 2
Remark. The C++ used is given in Appendix 1.

Fit to cubic was excellent as seen from MS Excel package:-

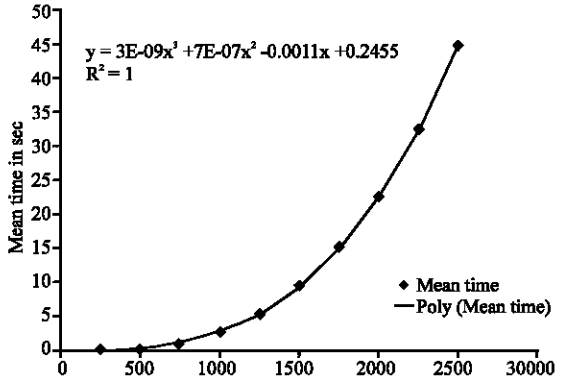


Fig. 3: Mean time in sec versus n

**Interpretation:** Evidently $T_{avg}(n)=O_{emp}(n^3)$ for the average case as well from the definition of empirical O. We again drop Kussmaul protection.

We next go into the third and last phase which is best case analysis (Fig. 3). For this the determinant's elements are selected as random integers from 0, 1, 2…n-1 and then all the elements of a randomly selected column are made zero.

**Phase 3: Best Case Analysis:**

System Specifications
Processor HP Pentium ® 4 CPU 3.00 GHz
Hard Disk 80 GB
RAM 448 MB
Operating System Windows XP Professional
Version 2002

Table 3: Best Case Complexity in Computing the Determinant of order n. All times are in seconds

| n | Trial 1 | Trial 2 | Trial 3 | Trial 4 | Trial 5 | Trial 6 | Trial 7 | Trial 8 | Trial 9 | Trial 10 | Mean Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 250 | 0.015 | 0.031 | 0.016 | 0.031 | 0.031 | 0.016 | 0.047 | 0.031 | 0.031 | 0.016 | 0.0265 |
| 500 | 0.25 | 0.25 | 0.079 | 0.187 | 0.156 | 0.265 | 0.11 | 0.266 | 0.25 | 0.141 | 0.1954 |
| 750 | 0.906 | 0.344 | 0.86 | 0.781 | 0.219 | 0.64 | 0.75 | 0.11 | 0.766 | 0.921 | 0.6297 |
| 1000 | 0 | 1.469 | 2.047 | 1.578 | 0.516 | 2.453 | 2.39 | 2.469 | 0.968 | 2.016 | 1.5906 |
| 1250 | 4.891 | 4.907 | 4.563 | 3 | 5.266 | 5.25 | 5.328 | 5.344 | 5.234 | 5.297 | 4.908 |
| 1500 | 3.61 | 5.25 | 9.235 | 9.204 | 9.297 | 9.266 | 0.562 | 8.14 | 6.093 | 9.406 | 7.0063 |
| 1750 | 1.422 | 7.672 | 5.39 | 15 | 10.422 | 6.062 | 7.86 | 7.203 | 14.453 | 15.203 | 9.0687 |
| 2000 | 6.172 | 11.797 | 8.094 | 19.203 | 22.297 | 22.578 | 22.5 | 22.75 | 3.938 | 0.453 | 13.9782 |
| 2250 | 10.375 | 11.86 | 26.266 | 27.828 | 32.828 | 22.016 | 31.672 | 3.282 | 24.953 | 30 | 22.108 |
| 2500 | 11.703 | 39.25 | 0.672 | 43.89 | 33.578 | 45.515 | 4.922 | 41.625 | 41.985 | 42.39 | 30.553 |

Table 4: Comparison table for algorithmic complexity bounds (mathematical vs statistical)

| Mathematical bounds | Statistical bounds |
|---|---|
| 1. Operations are *counted* | Operations are *weighed* |
| 2. A *separate* bound is provided for a *specific operation type* (mixing operations of different types is not permitted) | Weighing permits *collective consideration* of all operations for determining the bound (mixing operations of different types is very much permitted) |
| 3. Theoretically derivable | They are only *conceptual*. However their empirical estimates are certainly gettable in every system (the so called empirical O) |
| 4.They may be unrealistic at times | Guaranteed to be realistic |
| 5.They are *system independent* | They can only be *system invariant* (as suggested by identical empirical O in different systems) and may vary from system to system depending on the language and translator which may suit some specific operations and inputs |
| 6. They are ideal for analyzing worst case behavior | They are particularly suited to study average case behavior especially for complex codes |
| 7. They are exact | They are exact provided only they are system invariant |
| 8. They are trivial in comparison to the statistical bounds as it is easier to count the operations than to weigh them | They are non-trivial being based on operation weights. If weights are selected combining time with space the non-triviality of these bounds increases |
| 9. Determining a precise mathematical bound for a complex and arbitrary code can be a "deep intellectual challenge" (Aho *et al.*, 2000) | The basic methodology does not change no matter how arbitrary or complex the code may be. It is taken as a problem of designing and analyzing a computer experiment in which the response is a complexity such as time |

Service Pack 2
The C++ Code is given in Appendix 4.

**Interpretation:** Although there is a definite improvement in time for a given n (compare Table 3 with Table 1 and 2), we have $T_{best}(n)=O_{emp}(n^3)$. This is because although the algorithm had the facility for abrupt termination in case the determinant's value is zero, which was the case here, it is not clear in which step this can be detected especially for large n. That is to say, it is difficult to say how many times the loop over j will have to be executed which is crucial (see how the cubic pattern gets a little erratic initially before it stabilizes) and no probabilistic assumption including uniform seems to be realistic over the problem domain. Hence, our study is an eye opener which safeguards against making tall theoretical claims! Nevertheless as there is a definite saving in time, the facility is recommended. Of course when the first column has a null vector we invariably get an O(n) complexity as the outer j-loop executes once only and the inner i-loop executes n times (lines 120-140) and the condition "IF M=0..." is found true in line 150. This may be called "the very best of the best" case and is too optimistic to be of any practical use. If a row is zero or if two (or more) rows/columns are identical we again get $T_{best}(n)=O_{emp}(n^3)$ similarly (details omitted). Optimality design must be used

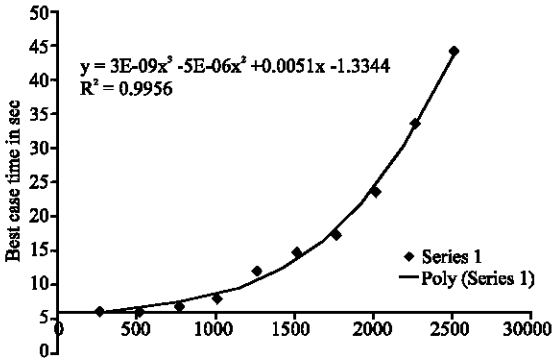


Fig. 4: Best case time in sec versus n

for prediction purpose. As a Kussmaul-protected quadratic fit did not compete well with a cubic fit we are skipping the details and would warn the reader from putting $T_{best}(n)=O_{emp}(n^2)$ or anything stronger which he/she might be tempted to put considering that the empirical O is after all only a subjective bound-estimate (Fig. 4).

This study establishes the "robustness" of statistical bounds for worst case, average case and best case complexity. While ideal for the average case the present study suggests in addition that there is no need to be

over-conservative in the worst case just as it safeguards against making tall optimistic claims for the best cases. In short the statistical bounds have a sense of "calculated guarantee" that is neither too risky nor too conservative. As promised, we summarize carefully the main differences between mathematical and statistical bounds in Table 4. Finally, we strongly recommend statistical bounds and their empirical estimates for every arbitrary algorithm that is deemed to be complex and even for a simple one if it is possible to challenge the probability distributional assumption over the problem domain.

Appendix 1: C++ Code for average case complexity:-

```
# include <iostream.h>
# include <iomanip.h>
# include <stdlib.h>
# include <time.h>
# include <math.h>
# include <conio.h>
# include <float.h>

void main()
{
      int i, j, row, k, m, g, count=0, n;
      float **a,ratio,temp,max;
  long double p;

randomize();

      cout << "Enter Size of Matrix: ";
      cin >>n;

      a = new float *[n];
  for (I=0;i<n;i++)
         a[i] = new float [n];
for (I=0;i<n;i++)
                {
       for (j=0;j<n;j++)
           {
                              //cin >> a[i][j];
                              a[i][j]=random(n);
           }
       }

/*for (I=0;i<n;i++)
  {
               for(j=0;j<n;j++)
                  {
                        cout << setw(10)<<a[i][j];
                  }
            cout <<endl;
       }*/

    clock_t start = clock();
          for(i=0;i<n;i++)
             {
                   max=a[i][i];
                   for(j=i;j<n;j++)
                        {
                              if (a[j][i]>max)
                                 {
                                    max=a[j][i];
                                    row=j;
                                 }
                        }
            if(max==0)
                {
        p=0;
                                    //cout << "\ndeterminant value "<<p<<
"terminated in column "<<I;
                                    //cout << "\ncount="<<count;
                                    goto stop;
                              }
          if(row!=i && a[row][i]==max)
                        {
                              for(k=0;k<n;k++)
                                    {
                                          temp=a[row][k];
                                          a[row][k]=a[i][k];
                                          a[i][k]=temp;
                                    }
                              count++;
                        }
           for (m=i+1;m<n;m++)
                {
                        ratio=a[m][i]/a[i][i];
                        for (g=0;g<n;g++)
                           {
                              a[m][g]=a[m][g]-(ratio*a[i][g]);
                           }
                }
      }

  /*cout << endl<<"matrix after upper triangularization"<<endl;
for (I=0;i<n;i++)
     {
            for (j=0;j<n;j++)
                {
          cout<<setw(10)<<setprecision(4)<<a[i][j];
}
cout << endl;
}*/

i=0;
j=0;
p=1;
  _control87(MCW_EM,MCW_EM);
while (i<n)
{
              p=p*a[i][j];
              I++;
                    j++;
              }
       if ((count%2)==1)
          p=p*(-1);

  stop:
  clock_t finish = clock();

//cout << "Determinant values="<<p;
//cout << endl<<"Count="<<count;

  double elapsed = float (finish - start)/CLK_TCK;
  cout <<endl<<"elapsed time in sec. "<<elapsed;
cout<<endl<<"Press anu key to continue....";getch();
}
```

Appendix 2: The Problem of Ill-conditioning

In some practical situations, we often deal with some systems of equations where small changes in the coefficients of the system produce a large change in the solution. Such systems are said to be *ill-conditioned*.

For an example, the system:

$$\left.\begin{aligned} 3x + y &= 2 \\ 3x + 0.01y &= 2.01 \end{aligned}\right\}$$

has the solution x = 1/3 and y = 1;
whereas the system:

$$\left.\begin{aligned} 3x + y &= 2 \\ 3.01x + y &= 2.04 \end{aligned}\right\}$$

has the solution x = 4 and y = -10.

Hence a small error in the coefficients of the system produced a large error in the solutions. Therefore, this system is ill-conditioned.

**3-Conditioned matrix:** If a system is ill-conditioned, its coefficient matrix is called *ill-conditioned matrix*. These kinds of matrices possess a very small determinant value (in comparison with its average element). Hence, it is also called as *near-singular matrix*.

**Examples of ill-conditioned matrices:**

- The coefficient matrix in the system of equations

$$\begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix} * \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}$$

  is ill-conditioned; because if we make a small change in the column vector on the right hand side of the equation, the actual solution i.e. $(1, 1, 1, 1)^t$ changes drastically. For example, if we make the column vector $(32.1, 22.9, 32.9, 31.1)^t$, the solution becomes $(6, -7.2, 2.9, -0.1)^t$.
- Hilbert matrix: The (i, j)th element of this matrix is equal to 1/(i+j-1). This is an ill-conditioned matrix. A Hilbert matrix of order 8 has a determinant value of order $10^{-31}$.

**Some measures of ill-conditioning:**
**Turing measure:**

Turing suggested two measures M(A) and N(A) for an ill-conditioned matrix A, defined as

- $M(A) = n \max|a_{ij}| * \max|a_{ij}|^{-1}$;

where, $|a_{ij}|$ and $|a_{ij}|^{-1}$ are the modulii of the (i, j)th element of the matrix A and its inverse respectively.

- $N(A) = n^{-1}\|A\| * \| A^{-1}\|$; where, $\|A\|$ is a norm of A. For example, one may consider $L_1$ or $L_\infty$ norm (Krishnamurthy and Sen, 1986).

**Neumann-Goldstine measure:**

Neumann and Goldstine (1947; see[4]) suggested the measure;

$$P(A) = \frac{|\lambda(A)|}{|\mu(A)|}$$

where, $\lambda(A)$ and $\mu(A)$ are the largest and the smallest magnitudes respectively of the latent roots of A.

**Measure based on linear dependence:**

3-Conditioning can also be measured by the inverse of the square-root of the smallest root of $A^tA$, since the ultimate cause of singularity is the linear dependence of its column (row) vectors.

The smallest root $\mu$ of $A^tA$ can be expressed as

$$\mu = \min \frac{x^t A^t A x}{\|x\|^2} = \min \frac{\|e\|^2}{\|x\|^2}$$

Where $e = a_1 x_1 + a_2 x_2 + - - - + a_n x_n$ ;

$a_i$'s being the column vectors constituting the coefficient matrix A, for all i and $x_n$'s are not all zero. Then ill-conditioning could be measured by $1/\mu$.

The matrix A of an ill conditioned system has the following symptoms:

(a) A small change in coefficients (matrix elements) causes a large change in the solution.
(b) Diagonal elements of the coefficient matrix tend to be smaller than off diagonal elements
© Computed Det(A) Det($A^{-1}$) deviates significantly from 1.
(d) Computed $(A^{-1})^{-1}$ deviates significantly from A.
(e) Computed $AA^{-1}$ deviates significantly from the identity matrix.
(f) Computed $(A^{-1})$ $(A^{-1})^{-1}$ deviates from the identity matrix more significantly than does $AA^{-1}$.

The best remedy of countering severe ill-conditioning is to increase the precision of computation. For polynomial fits, the simplest polynomial that fits the data well should be selected. Chebyshev polynomials are also made use of (Seber, 1987).

Appendix 3: C++ Code for worst case complexity:-

```
# include <iostream.h>
# include <iomanip.h>
# include <stdlib.h>
# include <time.h>
# include <math.h>
# include <conio.h>
# include <float.h>

void main()
{
      long int i,j,row,k,m,g,count=0,n;
      float **a,ratio,temp,max;
  long double p;
      randomize();

      cout << "Enter Size of Matrix: ";
      cin >>n;

      a=new float *[n];
  for (i=0;i<n;i++)
a[i]=new float [n];

      for (i=0;i<n;i++)
         {
                    for (j=0;j<n;j++)
                       {
                            //cin >> a[i][j];
                            a[i][j]=(float)rand()/RAND_MAX;
                       }
         }
  //cout.setf(ios::fixed,ios::floatfield);
  /*for (I=0;i<n;i++)
                    {
                            for(j=0;j<n;j++)
                           {
                                                cout << setw(10)<<a[i][j];
                                }
                    cout <<endl;
            }*/
  clock_t start = clock();
```

REST OF THE CODE IS IDENTICAL WITH APPENDIX 1

Appendix 4: C++ Code for best case complexity:-

```
# include <iostream.h>
# include <iomanip.h>
# include <stdlib.h>
# include <time.h>
```

```
# include <math.h>
# include <conio.h>
# include <float.h>

void main()
{
      int i, j, row, k, m, g, count=0, n, r;
      float **a,ratio,temp,max;
  long double p;

      randomize();

      cout << "Enter Size of Matrix: ";
      cin >>n;

      a=new float *[n];
  for (i=0;i<n;i++)
                  a[i]=new float [n];

  for (I=0;i<n;i++)
          {
                   for (j=0;j<n;j++)
                   {
                   //cin >> a[i][j];
                                                  a[i][j]=random(n);
                          }
          }

  r = random(n);
  for (I=0;i<n;i++)
     a[i][r]=0;

     /*for (I=0;i<n;i++)
        {
                  for(j=0;j<n;j++)
                      {
                                                  cout << setw(10)<<a[i][j];
                  }
                                        cout <<endl;
      }*/

  clock_t start = clock();
```

REST OF THE CODE IS IDENTICAL WITH APPENDIX 1


## ACKNOWLEDGEMENT

We thank Dr. Richard Graf and an anonymous referee for their suggestions (see InterStat June2007#4; http://interstat.statjournals.net/ where an initial draft is available online).


## REFERENCES


Aho, A., J. Hopcroft and J. Ullman, 2000. Data Structures and Algorithms, Pearson Education Reprint.

Chakraborty, S. and S.K. Sourabh, 2007. On Why an algorithmic time complexity measure can be system invariant rather than system independent. Applied Math. Computation, 190: 195-204.

Chakraborty, S., S.K. Sourabh, M. Bose and K. Sushant, 2007. Replacement sort revisited: The Gold Standard" unearthed!. Applied Math. Computation, 189: 384-394.

Krishnamurthy, E.V. and S.K. Sen, 1986. Numerical Algorithms, Affiliated East-West Pvt. Ltd., New Delhi.

Seber, G.A.F., 1987. Linear Regression Analysis, John Wiley.