Empirical Analysis of Linear System Solving Algorithms

Raquel M. de Souza [a], Anderson Zudio[a], Igor M. Coelho[a], Cristiane O. Faria[a], Paulo E. D. Pinto[a] and Fabiano S. Oliveira[a 1 2]

[a]Universidade do Estado do Rio de Janeiro, Brazil

Abstract

In this work, four different direct methods for solving large systems are implemented and also compared using a novel tool called EMA, in terms of accuracy versus number of required degrees of freedom, and CPU time. The tool provides an empirical analysis of a given algorithm (more precisely, of an implementation of such an algorithm provided as input) through iterative executions of growing complexity. Through the resource usage data collected on these executions, a function is devised by the tool to represent the resource consumption being analyzed. The studied methods are: a) Gaussian elimination, b) Gauss-Jordan elimination, c) LU decomposition, d) Crout Algorithm, considering the following matrix types: sparse, full, and the Hilbert matrix for the comparative tests, that can be obtained after being used some type of discretization modeling for problems in industry. For all methods, the rates of convergence predicted by theory have been found using EMA.

Keywords: Empirical Analysis of Algorithms, Large Systems, Gaussian Elimination, Gauss-Jordan Elimination, LU Decomposition, Crout Algorithm.

1. Introduction

Empirical analysis of algorithms is an important tool for discovering behavior where purely theoretical analysis fails to deliver results. Often enough, new methods for solving problems can follow patterns discovered during empirical analysis. As an important recent example, the primality problem being in P class, proved by Agrawal et. al [1], has started from empirical studies about conjectures and existing primality tests [2]. Empirical studies also support the validation of conjectures. In [3], an empirical

[1]E-mails: raquelmarcolino25@gmail.com, azudio@ime.uerj.br, igor.machado@ime.uerj.br, cofaria@ime.uerj.br, pauloedp@ime.uerj.br, fabiano.oliveira@ime.uerj.br (respectively)

[2]The first two authors were supported by CAPES. The last three authors were partially supported by FAPERJ.

study of the sorting algorithm Shellsort has been conducted by gathering data about multiple sequences to devise functions describing the behavior of the algorithm while using each sequence. These studies have gathered evidence in favor of a conjecture on the problem posed by Knuth [4].

However, the conduction of empirical studies is not straightforward. A great deal of laborious work, consisting of procedural and repetitive steps, is often involved. If enough attention is not dispensed by the experimenter, a failure in the conduction of some experimental step can ultimately compromise the concluding results. This type of error is in general hard to detect later on, and even harder to trace where it has happened. In this work, we describe EMA tool designed to mitigate those risks. Besides automatizing the conduction of experiments and presentation of results, the tool is also specialized in devising functions that represent the consumption of computational resources of the algorithm under analysis (as main examples, the time and space complexities of the algorithm). It does so by iterative executions of an implementation of such an algorithm on a set of inputs of growing complexity and monitoring and storing the consumption of used resources for each execution [5]. From the data obtained in the simulation, EMA works out its estimators.

In order to validate the use of the tool at obtaining consistent results to those predicted in theory, in this paper four algorithms for solving large linear systems were considered. In many computational simulation codes for real problems in science and engineering, solving large linear systems always appears as a step of the simulation that consumes more computational time. Therefore, proposals for innovative numerical algorithms are constantly being developed with purpose to obtain better performances in the resolution. Direct methods were the first to be developed and has been for years the most chosen because of its reliable in numerical behavior in obtaining the approximate solution [6, 7, 8]. In applications where the data demand more control in the numerical accuracy they are still being widely used [9, 10].

The four methods used here are: Gaussian Elimination, Gauss-Jordan Elimination, LU Decomposition, and Crout Algorithm with different types of matrices, aiming to contrast the theoretical asymptotic behavior of each algorithm and the function obtained by EMA. This paper is structured as follows: in Section 2, we summarize the methods tested in this work. In Section 3, we present more details of EMA tool. In Section 4, we show computational results on each method through the use of EMA. Finally, in Section 5, we present the concluding remarks.

2. Linear System Solving Algorithms

We are aiming to compare four different algorithms: Gaussian Elimination (GE), Gauss-Jordan Elimination (GJ), LU Decomposition (LU) and Crout Algorithm (CR) solving a linear system, $Ax = B$, where A is a non-singular square matrix of order n, x is the unknown variables vector, and B is the vector of constant terms. These well known methods have complexity of $\Theta(n^3)$. They work with the augmented matrix $M = [A|B]$, and the algorithm can be divided in 2 stages: Stage 1 intents to obtain an equivalent matrix M' in row echelon form, which represents a linear system of same solution but easier to obtain; and Stage 2 provides its solution directly[8, 6]. The former stage is commonly named as Forward Elimination method, which receives M and n as input and outputs M', doing $\frac{n^3}{3}$ multiplication operations plus $\frac{n^3}{3}$ addition operations as stated in Burden and Faires [7]. Following, we summarize the main characteristics of the four methods:

- Gaussian Elimination: the first stage of this method is composed by Forward Elimination. The second stage consists of solving the last equation from M', and then obtaining each further solution by bottom-up substitutions of previously calculated variables. That is $0.5n$ multiplications plus $0.5n$ additions in n equations, totalizing n^2 arithmetic operations. Therefore, Stage 2 has complexity of $\Theta(n^2)$.

- Gauss-Jordan Elimination: it starts with Forward Elimination followed by Backward Elimination method with M' as input which works similarly to Forward Elimination and outputs a matrix M'' in canonical form. The latter method is similar to Forward Elimination taking a bottom-up approach to achieve its goal in $\frac{n^3}{3}$ arithmetic operations totalizing n^3 operations for the entire stage. Stage 2 was not considered in this work, because the solution will be B itself.

- LU Decomposition: it splits A into two square matrices, $A = LU$, where L and U are lower and upper triangular matrices respectively. Considering $LUx = B$ and $Ux = y$, Stage 2 consists of first solving the linear system $Ly = B$ by top-down substitution, then followed by solving $Ux = y$ by bottom-up substitution. Therefore, this stage has $2n^2$ arithmetic operations.

- Crout Algorithm: it works like LU, but the main difference is: U will have its main diagonal values equal to one. That is, at the beginning

of Stage 1 we copy A's main diagonal into L. Then it proceeds with Forward Elimination.

3. EMA

EMA (standing for *EMpirical Analysis of algorithms*) is a novel tool created to empirically estimate resource complexity (time and space, as main examples) of an algorithm. EMA works by receiving as input both an implementation of such an algorithm (an executable program), and another program specialized in generating inputs with varying sizes for the former. Given that, EMA uses this input-generator to create inputs with growing complexity (in terms of resource consumption) and, in the sequence, runs iteratively the algorithm using those inputs. Once data have been gathered during simulation, EMA employs nonlinear regression to devise functions that best represent the consumption behavior of those resources under analysis.

In this work, the input-generator program receives as input a natural n, standing for the order of the matrix to be built. Besides, we are interested in EMA analysis of total running time and total space (RAM usage). In general, EMA can suggest the range of specific values of n to be used in simulation, through a process called calibration, restrict to user-defined time and memory maximum usages, or users can define such values for themselves.

For each specific value of n, EMA runs the input-generator followed by the algorithm on the generated input and collects data regarding, in our case, running time and memory allocation. Using the model function

$$R(n) = a_0 n^{a_1} a_2^{n^{a_3}} \log_2^{a_4} n \log_2^{a_5} n, \text{ with } a_0,\, a_2 > 0$$

EMA uses nonlinear regressions to adjust the parameters a_i for all $1 \leq i \leq 5$ to fit the gathered data of $R(n)$, the resource consumption of a matrix of order n. The goal of the fitting procedure is to minimize the sum of the quadratic errors (for each value, the error is the difference between measured value and that given by estimation). EMA reports multiple functions for each resource, grouped as follows:

- **Minimum Error (ME):** this function has the smallest mean squared error [11]. The error of each other function is reported both in terms of absolute value and as relative percentage of the ME function error.
- **Feasible to tie break:** functions that can differ in 5% from the ME function in non-tested values, apparently viable to be verified given

the current computational maximum resources. The smallest of such values is called *tie breaker*.

- **Unfeasible to tie break:** in contrast to the previous class, a function in this group is considered that if it can differ in 5% from the ME function in non-tested values, those values are impractically big. That is, the tie breaker value is unfeasible to be tested given the current resource limitation.

- **Equivalent:** the sum of squared errors of these functions in comparison to that of the ME function is at most 5%. A function in this group approximates very nearly the ME function and, for practical purposes, being considered equivalent to the ME function in the tested range of values. Those functions can be also be classified as feasible or unfeasible to tie break.

- **Best:** a singleton group, representing the function that EMA chooses as its best guess of the actual function which determines the resource consumption (note that, according to the previous groups, there may be several closely-related functions). EMA chooses a function from the equivalent group having the fewest non-neutral values for parameters[†] and, in case of ties, the one having the smallest error. The motivation is to choose, among all functions equivalent to the ME function, the simplest one (Occam's Razor).

After presenting all functions categorized on the above groups, EMA finishes its operation. EMA has proved to be very effective and convenient for the analysis that has been conducted in this work.

4. Computational Experiments

By solving single instances of linear systems with variable degrees of freedom, we will present an empirical comparative test of the four methods presented in Section 2. Multiple sets of tests were generated, each one is composed by the following matrix types: sparse, full, and Hilbert matrix. Every matrix was identified by its order n, where $2 \cdot 10^4 \leq n \leq 5 \cdot 10^4$, and every test set was studied individually for each algorithm. EMA was responsible for collecting and analyzing test data and the configuration established for it was to use a previously generated matrix from the set corresponding

[†]A neutral value to a parameter is such a value that does not change the function evaluation if the parameter is removed from the function, such as a value of 1 in a multiplication or of 0 in an exponent.

to its order n, ensuring that every method solved the same linear system for the comparative test.

Every method was implemented using *C++* programming language and time marks were placed to acquire Stage 1 and Stage 2 running time, while the application was in charge of counting the total number of arithmetic operations performed during each algorithm. EMA was in charge of collecting total running time and memory usage. Every test was performed with the following specifications: Intel® Core ™ i7-4820K Processor 3.7 GHz; 16GB of RAM; Ubuntu 14.04 LTS.

After completing each simulation, EMA provided multiple functions to describe the behavior of each resource using every point obtained. Tables 1 and 2 present every *Best* function devised by EMA for each test set with its following complexity.

Table 1: Best Functions provided by EMA.

Resource	Full	Sparse	Hilbert	Complexity
Gaussian Elimination				
Total Time (ns)	$1.32\ n^3$	$1.32\ n^3$	$1.31\ n^3$	$\Theta(n^3)$
Stage 1 (ns)	$1.32\ n^3$	$1.32\ n^3$	$1.31\ n^3$	$\Theta(n^3)$
Stage 2 (ns)	$2.59\ n^2$	$2.58\ n^2$	$2.56\ n^2$	$\Theta(n^2)$
Space (Bytes)	$7.89\ n^2$	$7.89\ n^2$	$7.89\ n^2$	$\Theta(n^2)$
Gauss-Jordan Elimination				
Total Time (ns)	$1.98\ n^3$	$1.98\ n^3$	$1.98\ n^3$	$\Theta(n^3)$
Stage 1 (ns)	$1.98\ n^3$	$1.98\ n^3$	$1.98\ n^3$	$\Theta(n^3)$
Space (Bytes)	$7.89\ n^2$	$7.89\ n^2$	$7.89\ n^2$	$\Theta(n^2)$
LU Decomposition				
Total Time (ns)	$1.84\ n^3$	$1.84\ n^3$	$1.84\ n^3$	$\Theta(n^3)$
Stage 1 (ns)	$1.84\ n^3$	$1.84\ n^3$	$1.84\ n^3$	$\Theta(n^3)$
Stage 2 (ns)	$5.38\ n^2$	$5.38\ n^2$	$5.37\ n^2$	$\Theta(n^2)$
Space (Bytes)	$23.5\ n^2$	$23.5\ n^2$	$23.5\ n^2$	$\Theta(n^2)$
Crout Algorithm				
Total Time (ns)	$1.84\ n^3$	$1.84\ n^3$	$1.84\ n^3$	$\Theta(n^3)$
Stage 1 (ns)	$1.84\ n^3$	$1.84\ n^3$	$1.84\ n^3$	$\Theta(n^3)$
Stage 2 (ns)	$5.35\ n^2$	$5.35\ n^2$	$5.35\ n^2$	$\Theta(n^2)$
Space (Bytes)	$23.5\ n^2$	$23.5\ n^2$	$23.5\ n^2$	$\Theta(n^2)$

Table 2: Best Functions for Basic Operations provided by EMA.

Resource	Full	Sparse	Hilbert	Complexity
Gaussian Elimination				
Total	0.66 n^3	0.66 n^3	0.66 n^3	$\Theta(n^3)$
Stage 1	0.66 n^3	0.66 n^3	0.66 n^3	$\Theta(n^3)$
Stage 2	1.00 n^2	1.00 n^2	1.00 n^2	$\Theta(n^2)$
Gauss-Jordan Elimination				
Total	1.00 n^3	1.00 n^3	1.00 n^3	$\Theta(n^3)$
Stage 1	1.00 n^3	1.00 n^3	1.00 n^3	$\Theta(n^3)$
LU Decomposition				
Total	0.66 n^3	0.66 n^3	0.66 n^3	$\Theta(n^3)$
Stage 1	0.66 n^3	0.66 n^3	0.66 n^3	$\Theta(n^3)$
Stage 2	2.00 n^2	2.00 n^2	2.00 n^2	$\Theta(n^2)$
Crout Algorithm				
Total	0.66 n^3	0.66 n^3	0.66 n^3	$\Theta(n^3)$
Stage 1	0.66 n^3	0.66 n^3	0.66 n^3	$\Theta(n^3)$
Stage 2	2.00 n^2	2.00 n^2	2.00 n^2	$\Theta(n^2)$

Our GE and GJ implementation uses a matrix of `double` (8 Bytes real number) to store the input. Therefore, their space usage should be close to $8n^2$ Bytes. However, LU and CR needs two more matrices to store L and U, thus the space usage for them should be close to $24n^2$ Bytes, that is, we have a theoretical ratio of 3 between memory usage of those two groups. Table 1 shows equivalent functions for space usage, $7.89n^2$ and $23.5n^2$ respectively, with ratio 2.98. As for Stage 2, GE solves n equations by substitution, whereas LU and CR solves $2n$ equations. According to Table 1, EMA provides 2.59 and 5.38 coefficient for Stage 2 running time meaning the ratio 2.08 is equivalent, as expected.

Table 2 shows information regarding arithmetic operations being consistent with the theory presented in Section 2 across all stages. Table 1 presents consistent rates of convergence predicted by theory. All simulations and analysis were independent and EMA devised equivalent functions to each method of its respective resource across all types of input utilized, concluding that analysis made by EMA is not sensible to input. Figures 1, 2, and 3 displays each acquired point from the simulation done by EMA of each method with full and Hilbert matrices test set. They also show some of the *Best* functions obtained from Table 1 proving they fit the points. All results proves that EMA is robust.

According to Table 1, each coefficient obtained for total running time exhibits an important information about the performance of each algorithm at solving single instances of linear systems: GE is the fastest method, being 50% faster than GJ, and 39.4% faster than both LU and CR which had the same overall performance. As previously observed, LU and CR uses more memory than GE and GJ.

5. Concluding Remarks

This work presented EMA, a novel tool capable of obtaining functions that represent resource usage complexity of an algorithm through automatic simulation of an implementation of it. In our tests, EMA was robust in validating the problem of solving large linear systems by direct methods, in the sense that EMA's results was consistent to those predicted by theory across different input types for the following four methods: Gauss Elimination (GE), Gauss-Jordan Elimination (GJ), LU Decomposition (LU) and Crout Algorithm (CR). As an indirect important consequence, this work compares the practical performance of those methods at solving the same set of instances of linear systems. The comparative test not only showed that GE is 50% faster than GJ, and 39.4% than LU/CR, but it also showed that GE/GJ has less memory usage than LU/CR. In our experience, EMA demonstrated to be a convenient way to automate the conduction of computational experiments and to analyze the asymptotic behavior of an algorithm.

References

[1] Agrawal, M., Kayal, N., Saxena, N. PRIMES is in P, 2004, JSTOR.

[2] Bornemann, F. PRIMES Is in P: A Breakthrough for" Everyman", 2003, American Mathematical Society.

[3] Weiss, Mark Allen. Short Note – Empirical study of the expected running time of Shellsort, 1991, Br Computer Soc.

[4] Knuth, Donald E. The Art of Computer Programming, Vol. 3, Sorting and Searching, 1973, Addison-Wesley, Reading, Mass.

[5] Oliveira, F. S. EMA – WebPage (2016), `http://fabianooliveira.ime.uerj.br/ema`, 2016, Last access: September 23, 2016.

[6] Roger, Teman. Algèbre Linéaire, C3 Analyse Numérique, 1969-1970, University of California.

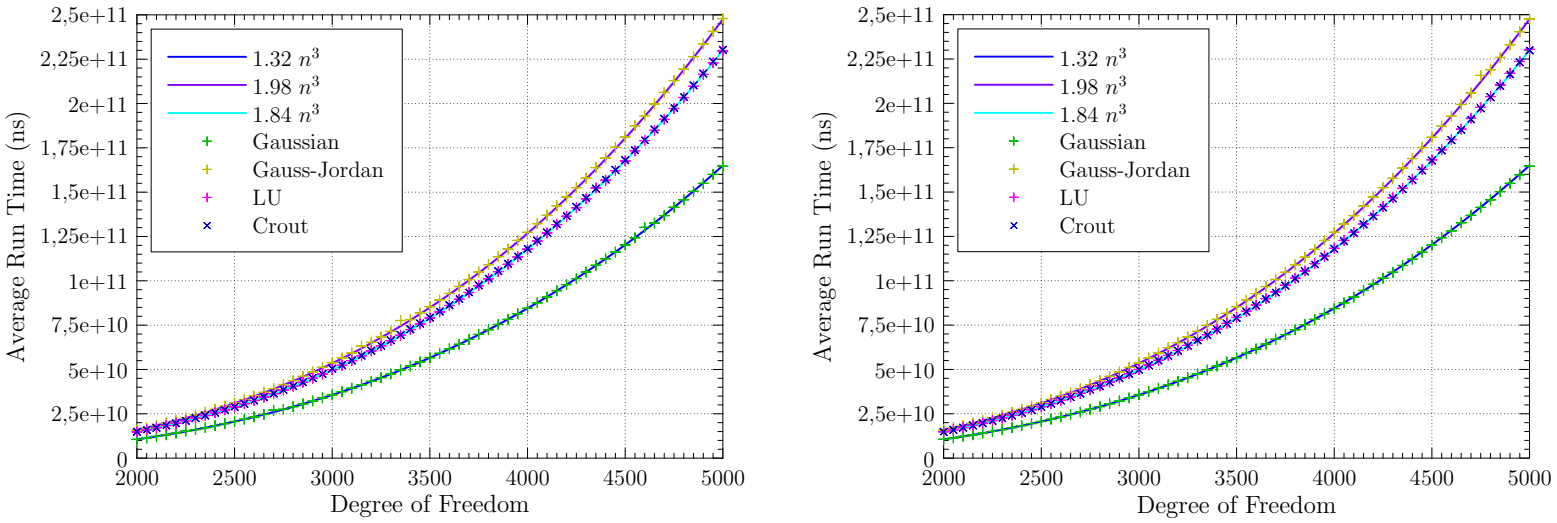
[7] Burden, Richard L., Faires, J. Douglas. Numerical analysis, 2004, 8th edition, Brooks/Cole.

[8] Golub, Gene H., Van Loan, Charles F. Matrix computations, 1989, The Johns Hopkins University Press.

[9] Cortes, A. M. A.,Coutinho, A. L. G. A., Dalcin, L., Calo, V. M. Performance evaluation of block-diagonal preconditioners for the divergence-conforming B-spline discretization of the Stokes system, 2015, Journal of Computational Science.

[10] Devakar, M., Ramesh, K., Chouhan, S., Raje, A. Fully developed flow of non-Newtonian fluids in a straight uniform square duct through porous medium, 2016, Elsevier.

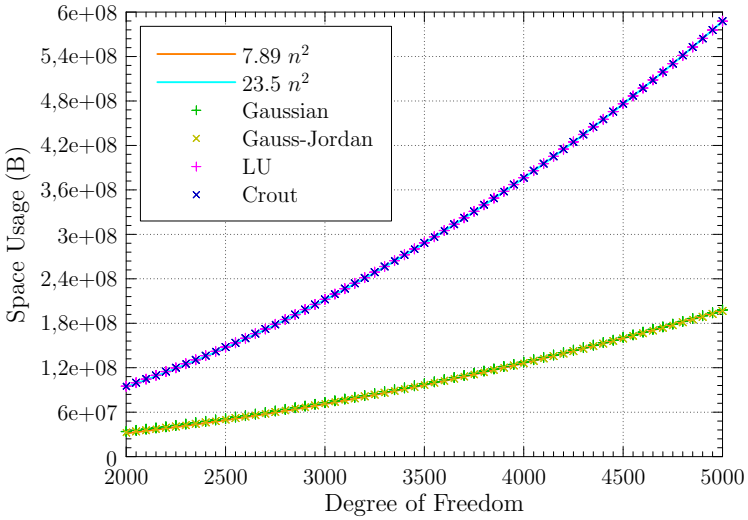
[11] Lehman, E. L., Casella, G. Theory of point estimation, 1991, Wadsworth & Brooks/Cole Advanced Books & Software.



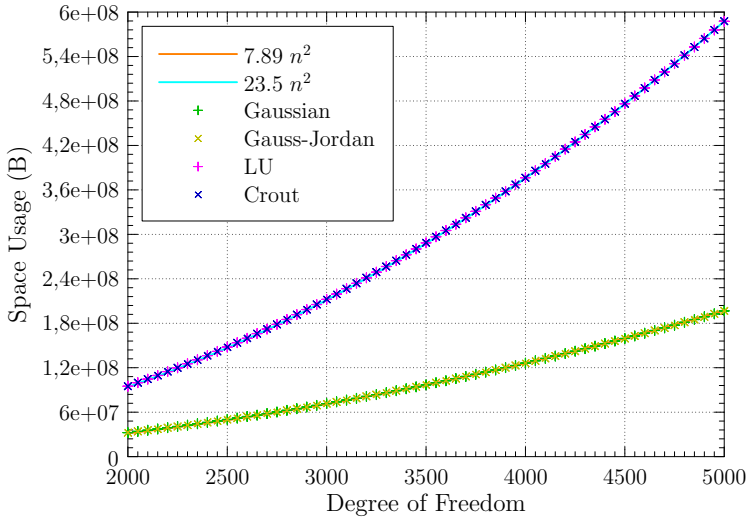
(a) Full matrix.

(b) Hilbert matrix.

Figure 1: Stage 1 running time versus number of required degrees of freedom

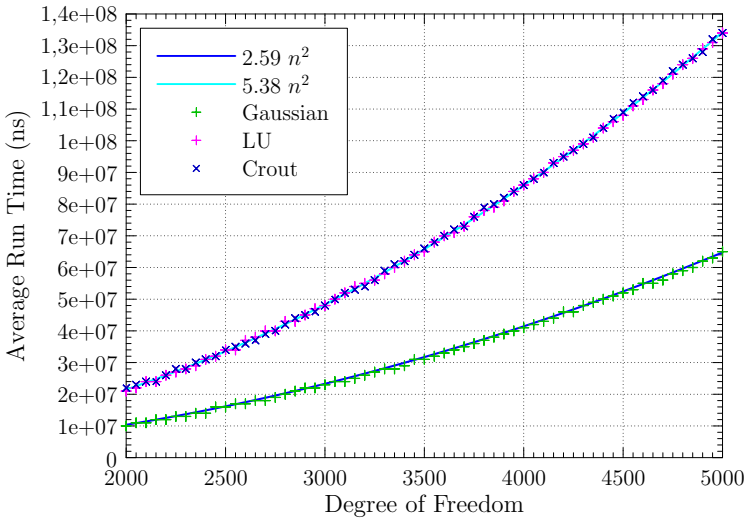


(a) Full matrix.

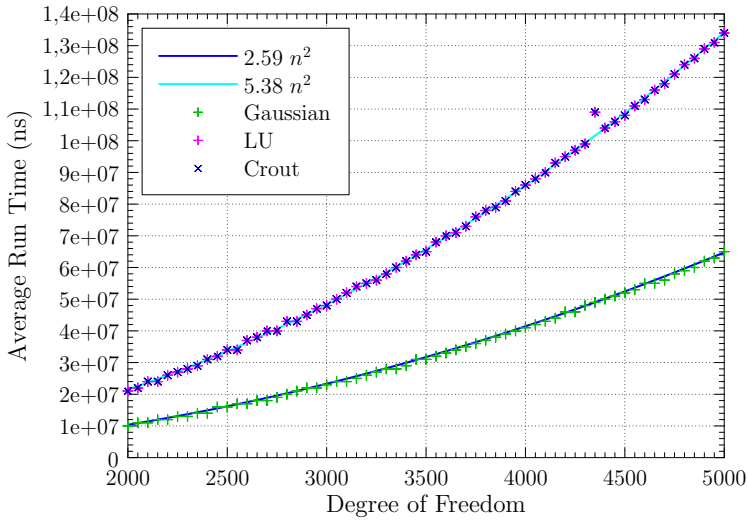


(b) Hilbert matrix.

Figure 2: Memory usage versus number of required degrees of freedom



(a) Full matrix.



(b) Hilbert matrix.

Figure 3: Stage 2 running time versus number of required degrees of freedom