Annales Univ. Sci. Budapest., Sect. Comp. 49 (2019) 219–227

COMPARING COMPUTATIONAL SPEED OF MATLAB AND MATHEMATICA ACROSS A SET OF BENCHMARK NUMBER CRUNCHING PROBLEMS

Michael Freitas Gustavo and János Tóth

(Budapest, Hungary)

Communicated by László Szili

(Received March 31, 2019; accepted July 21, 2019)

Abstract. MATLAB and *Mathematica* are the most widely used mathematical programming tools, each with its associated stereotypical advantages over the other. In order to test these stereotypes and determine where these advantages truly lie, a set of low-level benchmarking tests, covering a wide range of numerical and symbolic problems, were constructed and timed in each language. It was determined that *Mathematica* still enjoys an enormous advantage in the solution of symbolic problems, but MATLAB is better able to import and solve problems which involve large amounts of numerical data.

1. Introduction

Since the first appearance of mathematical program packages*, they have been being compared. Nowadays it is more frequent to find informal comparisons in forums on the internet [8, 9, 11, 13, 1, 10], even Wikipedia has a good systematic comparison of the existing systems [12].

Key words and phrases: CAS, speed comparison, *Mathematica*, MATLAB.
2010 Mathematics Subject Classification: 65-04.
The Project is supported by the funding from the National Research Development and Innovation Office (Grant number: SNN T25739). The authors would also like to acknowledge the assistance of O. Sáfár and the remarks and criticisms of the reviewer.

*Although algebra pays an important role in creating them, computer algebra is a misleading name for such software; we prefer mathematical program package.

Perhaps the two most popular mathematical software packages used in academia today are *MathWorks'* MATLAB and *Wolfram's Mathematica*. Both systems cover much of the same scope with wide functionality extending from symbolic programming, differential and integration problems, 3D image generation and algebraic manipulations. Of course each has their associated areas of strength; MATLAB is often favoured within the engineering community for its perceived advantages at numerical calculation, while *Mathematica* is utilised by physicists and mathematicians for its symbolic processing abilities [8].

In fact this is a common (and persistent) stereotype surrounding both products, and one that is often cited by users who strongly favour one program over the other. Given the fact that this generalisation has been around for many years, and that the packages themselves have been frequently updated, it is possible that the numerical and symbolic processing speeds of both programs have improved significantly in that time.

To that end, in this paper both programs (MATLAB R2018b Update 3 and *Mathematica* 12.0) are speed tested in a variety of symbolic and numerical problems. The purpose of this is not to further inflame the rivalry between proponents of either program, but to help users make informed decisions about their software choices.

2. Methodology

Both packages include built-in benchmarking scripts which, although originally designed as a way to compare performance between machines, provide a set of low-level tests which are crucial to any package. These codes are thus ideal for comparing the number-crunching abilities of each language. The tests used for the benchmarking comparison were adapted from the built-in *Mathematica* benchmarking notebook (`WolframMark Report.nb`), generated by running `BenchmarkReport[]` from the `Benchmarking` package. This script was selected because the MATLAB benchmarking program did not make its code available and consists only of six tests, two of which focus on image processing and the remainder had significant overlap with the *Mathematica* file.

The core functionality of each of the fifteen tests in the original *Mathematica* benchmark were left largely unchanged but several important modifications were made:

1. Several tests included steps to randomly generate a set of numbers. In order to ensure repeatability and fair comparison between the packages, these procedures were replaced with references to a data file. This CSV data file, generated by *Microsoft Excel 365* contains 1 048 576 randomly generated numbers (each with 6 decimal precision) from a [0,1] uniform

distribution. This file is read in once by both packages into a `data` variable at the beginning of the run. This process is also timed and forms 'Test 0 : Import Data' below.

2. According to the recommendations of McLoone [6] `Module`, `Block` and `With` were interchanged where possible to help ensure the fastest results. In another effort to save time, the `data` variable was explicitly flagged as a set of real values; *Mathematica* treats all inputs as complex unless otherwise stated. For similar reasons integer values were switched to floating point numbers. It is of interest to note, however, that the effect of these changes was very marginal and often negligible.

3. In order to reproduce the tests exactly in MATLAB, some tests[†] had to be simplified so that the answer did not exceed its maximum number value (1.7977×10^{308}). *Mathematica*'s maximum number is theoretically unlimited, but is a function of the computer system being used; for this work the maximum number was $1.605\,216\,761\,933\,662 \times 10^{1\,355\,718\,576\,299\,609}$.

4. Finally, in terms of timing methodology, each test was measured independently using MATLAB's `timeit` function or *Mathematica*'s `RepeatedTiming` function. Both functions ran the test multiple times, the first returned the median time taken, the second returned the mean of the times in the second and third quartiles. Both methods use wall-clock time and the repeated nature of the timings help avoid first-time costs and obviously add statistical significance. Although the approaches of the two are slightly different we accept them as comparable. To focus the timing on the important elements of each test only the core operation was timed, preparatory operations like reshaping the data into matrices was excluded.

The modified *Mathematica* code was replicated in MATLAB as closely as possible by using corresponding built-in functions where available. In the few cases where this could not be done, the code was written as efficiently as possible. The source code used in each program is included in [2].

In total sixteen tests were conducted; below follows a short description of each:

Test 0: Data Importing Reads numbers from a CSV into a variable for use in the other tests.

Test 1: Data Fitting Finds the optimal parameters a, b and c which fit the function

$$f(x, y, z) := \ln(ax) - \left| \frac{\cos\left(\frac{z}{b}\right)}{cy} \right|,$$

[†]Specifically, 'Test 6: Gamma Function' and 'Test 7: Large Integer Multiplication'.

from a 4D array of x, y, z and $g(x,y,z)$ values where $x,y,z \in \{0.2k | k = = 1,2,3,\ldots,44\}$ and

$$g(x,y,z) := \ln(120x) - \left| \frac{\cos\left(\frac{z}{300}\right)}{140y} \right|.$$

Test 2: Digits of Pi Finds and lists the first one million digits of π.

Test 3: Discrete Fourier Transform Finds the discrete Fourier transform of a vector and repeats the process eleven times.

Test 4: Eigenvalues of a Matrix For a a 420×420 matrix and b a diagonal matrix containing a 420 element column vector (both are extracted from `data`), then this finds the eigenvalues of matrix m, where $m := a \cdot b \cdot a^{-1}$. The process is repeated six times.

Test 5: Elementary Functions Applies e^x, $\sin x$ and $\tan x$ to each element of `data` sixty times each.

Test 6: Gamma Function `data` is adjusted to range between [160, 170] and the Gamma function of each element is found 1 000 times.

Test 7: Large Integer Multiplication `data` elements are first multiplied by 1000, cubed and then rounded to integer values. The *Mathematica* default is to round the midpoint to the nearest even number. In MATLAB this is done with the `convergent` function. This vector is then reshaped into two column vectors of 524 288 elements[‡]. Each corresponding pair of elements is then multiplied by one another.

Test 8: Matrix Arithmetic Reshapes a subset of `data` into an 840×840 matrix m and applies $f(x) := (1 + 0.5x)^{127}$ to each of its elements.

Test 9: Matrix Multiplication Performs matrix multiplication on two identical 1024×1024 matrices constructed from `data` and repeats this twelve times.

Test 10: Matrix Transpose Finds the transpose of `data`, reshaped into a 1024×1024 matrix, eighty times.

Test 11: Numerical Integration Finds the numerical integral of $f(x,y) := := \sin\left(x^2 + y^2\right)$ where $x, y \in [-2.6\pi, 2.6\pi]$.

[‡]Note that the two packages are reshaped slightly differently in this case ($2 \times 524\,288$ in *Mathematica* and $524\,288 \times 2$ in MATLAB) simply to better fit the actual multiplication step. Since these reshaping steps are not included in the timing, the differences are negligible.

Test 12: Polynomial Expansion Symbolically evaluates the expression

$$f(x) := \prod_{c=1}^{350} (x + c)^3.$$

Test 13: Random Number Sort Multiplies each element of `data` by 50 000 and then sorts the vector, from smallest to largest, fifteen times.

Test 14: Singular Value Decomposition Finds the Singular Value Decomposition of `data` reshaped into a 1024×1024 matrix.

Test 15: Solving a Linear System Finds x in the linear equation $m \cdot x = v$ where m is `data` reshaped into a 1024×1024 matrix and v is a vector filled with the values of `data` between indices 51 814 and 52 837.

To compare the results of the two packages, every element of each test result was exported to CSV and then compared in *Microsoft Excel 365* by taking the difference between each result and then finding the minimum, maximum and average of the differences.

3. Results and discussion

Test results were compared for equality before analysing the timings. For most tests (specifically tests 4, 5, 6, 7, 8, 9, 13, 14 and 15) any differences occurred, on average, at the fifteenth significant figure. This coincides with the sixteen decimal precision of MATLAB [5], fifteen decimal precision of *Excel* [7], and the machine precision of *Mathematica* which, on the tested computer, was sixteen. Thus these differences can be attributed to rounding and approximation errors and can be considered identical otherwise. Comparision of the remaining tests was less straightforward and are summarised in Table 1.

Table 2 lists the timings of each test which are also represented graphically in Figure 1. To make the comparison clearer, Figure 2 shows the difference in time for each test. Generally MATLAB had a significant advantage over *Mathematica* particularly in the speed at which it imported data, transposed matrices, calculated gamma functions, and sorted numbers. It is not clear why the *Mathematica* algorithms were particularly slow in these regards. On the other hand, *Mathematica* had an enormous advantage in symbolic processing (not surprising given that this is its main function) and element-wise matrix operations (which turned out quicker for *Mathematica* despite this perhaps being MATLAB's core functionality). The differences between the other tests were small but generally swung in favour of MATLAB.

Test	Comparison of Results
1	MATLAB: $a = 120.000000005858$, $b = 139.999999017496$, $c = = 299.196900459061$ *Mathematica*: $a = 120$, $b = 140$, $c = -300$. We note that because c is in the argument of the cosine function the sign does not matter. It is unreasonable to expect identical answers from a data fitting test. For the purposes of this work we consider the answers to be identical.
2	Due to the maximum cell content limit of *Excel*, only the first 32767 digits could be compared but were found to be identical.
3	These answers differed moderately in three key ways but remained comparable. First, the complex argument was consistently opposite in sign between the packages. Secondly, MATLAB's answers were consistently 1000 times larger than *Mathematica*'s. Thirdly, the numbers differed at the third or fourth significant figure. Since this function is non-unique we accept these differences.
10	Both packages transposed the matrices exactly.
11	MATLAB: 3.14741405891796 *Mathematica*: 3.14741405930589 It is unreasonable to expect identical answers from a numerical integration test. For the purposes of this work we consider the answers to be the same.
12	All coefficients of the constructed polynomial were found to be identical, this is because this symbolic test uses variable precision in MATLAB and is thus not limited by the floating point precision of other tests.

Table 1. Tests with non-straightforward comparisons

It is pertinent to note the difference between numeric and symbolic calculation here. MATLAB is primarily a numerical platform which uses numbers with floating point precision [3], however, it does provide symbolic and variable precision tools to perform numeric calculations with an arbitrary number of significant figures [4]. The maximum number of significant figures in these regimes are 2^{29}, but the absolute largest number is still 1.7977×10^{308}. To use these tools, they must be explicitly invoked in the calculation. On the other hand, *Mathematica* is also equipped with both numeric and symbolic capabilities but the choice of which to use is left to the system depending on the input number of significant figures [14]. The difference between the two types of calculation can have a profound effect on calculation speed [4]. To ensure that *Mathematica* was computing numerically rather than symbolically, each test result was checked for its final precision. Only Test 2: Digits of Pi and

Test		Time (s)	
		MATLAB	*Mathematica*
0	Data Importing	0.4384	2.7000
1	Data Fitting	0.7326	1.6100
2	Digits of Pi	0.0225	0.0000
3	Discrete Fourier Transform	0.2152	0.7390
4	Eigenvalues of a Matrix	0.6122	0.6600
5	Elementary Functions	1.2814	2.7700
6	Gamma Function	1.2798	4.7100
7	Large Integer Multiplication	0.4641	0.0077
8	Matrix Arithmetic	3.0249	1.0800
9	Matrix Multiplication	0.8696	1.2000
10	Matrix Transpose	0.4702	5.1160
11	Numerical Integration	0.1270	0.8200
12	Polynomial Expansion	6.1698	0.0005
13	Random Number Sort	0.8183	3.0800
14	Singular Value Decomposition	0.4544	0.9700
15	Solving a Linear System	0.7018	0.9990
	Total	17.6822	26.4622

Table 2. Timing results of sixteen numerical and symbolic test problems on MATLAB and *Mathematica*

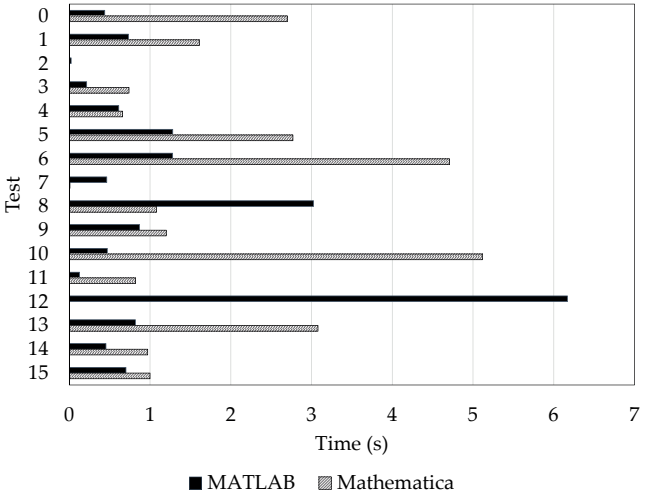

Figure 1. Testing times in each language for each test

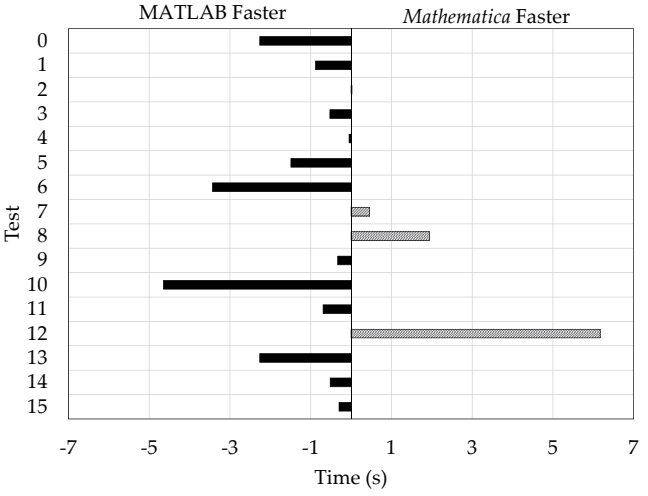

Figure 2. Difference in testing times in each language for each test

Test 12: Polynomial Expansion did not have machine precision (i.e. were calculated using symbolic tools). This matches the behaviour of MATLAB and indicates that our comparisons are fair and appropriate.

4. Conclusion

The computational time of *Mathematica* and MATLAB, across a wide range of low-level computationally intensive problems, was compared. It was determined that significant differences still exist between the languages in terms of their strengths and appropriate areas of application. MATLAB was unable to compete with *Mathematica*'s symbolic programming advantages. On the other hand, MATLAB demonstrated clear and repeated computational speed advantages in most numeric problems. The task of comparing two packages, however, is harder than it would appear at first glance due to core differences in the structure of the programs, the manner in which results are calculated, and the number of significant figures used.

References

[1] **Abbasi, N.M.,** Comparing Matlab, Mathematica and Maple numerical speed for matrix rank calculation, September 2, 2016. https://12000.org/my_notes/rankTest/test.htm

[2] **Gustavo, M.F. and J. Tóth,** Codes and data,
http://ac.inf.elte.hu/Vol_049_2019
[3] **MathWorks,** Floating-Point Numbers,
https://www.mathworks.com/help/matlab/
[4] **MathWorks,** Choose Symbolic or Numeric Arithmetic,
https://www.mathworks.com/help/symbolic/
[5] **MathWorks,** Increase Precision of Numeric Calculations,
https://www.mathworks.com/help/symbolic/
[6] **McLoone, J.,** Wolfram Blog: 10 Tips for Writing Fast Mathematica Code, December 7, 2011.
https://blog.wolfram.com/2011/12/07/
[7] **Oppenheimer, D.,** Understanding Floating Point Precision aka "Why does Excel Give Me Seemingly Wrong Answers?", April 10, 2008.
https://www.microsoft.com/
[8] **ResearchGate,** Is MATLAB or Mathematica more appropriate for mechanical engineers?, 2014.
https://www.researchgate.net/post/
[9] **ResearchGate,** Which one is faster? Matlab or Mathematica?, 2017.
https://www.researchgate.net/post/
[10] **StackExchange: Mathematica,** Dramatic speed difference of code on Matlab and Mathematica,
https://mathematica.stackexchange.com/questions/74108/
[11] **Quora** Why isn't Mathematica not as popular as MATLAB or Python?,
https://www.quora.com/
[12] **Wikipedia,** List of Computer Algebra Systems,
https://en.wikipedia.org/
[13] **Wolfram Community,** Matrix operation speed: Mathematica vs Matlab?,
https://community.wolfram.com/groups/-/m/t/864778
[14] **Wolfram,** Numerical Precision,
https://reference.wolfram.com/language/tutorial/

M. F. Gustavo and J. Tóth
Department of Analysis
Budapest University of Technology and Economics
Budapest
Hungary
mgus93@gmail.com
jtoth@math.bme.hu