PERFORMANCE IN SCILAB-5.3.0 VS SCILAB-5.3.0-SSE3

Ver.	Date	Author	Change Description	Baseline Version
1.0	30/01/2011	Ronit Ben Shalom	-	1.0

Distribution: Public.

TABLE OF CONTENTS

1. INTRODUCTION

In this document, we analyze the performance in Scilab-5.3.0 versus Scilab-5.3.0-SSE3. We present the functions to measure the performance of an algorithm and our conclusions.

The benchmark is set of Scilab Math core functions. It is performed on Scilab v5.3.0/5.3.0-SSE3 under Windows XP 64 bits. The processor is an Intel Core 2 Duo E7500 at 2.93 GHz.

We use Intel Math Kernel Library (MKL) – numerical library which provide optimized linear algebra features.

2. MEASURING THE PERFORMANCE

In this section, we describe the functions which allow measuring the time required by a computation.

We present the *tic*, *toc* and *timer* functions.

tic, toc

In order to measure the time required by a computation, we use the *tic* and *toc* functions which measure the user time in seconds. The user time is the wall clock time, that is, the time that was required by the computer from the start of the task to its end. This includes the computation itself, of course, but also all other operations of the system, like refreshing the screen, updating the file system, letting other process do a part of their work, etc...

The call to the *tic* function starts the counter and the call to the *toc* function stops it, returning the number of elapsed seconds.

This time measure is not very reliable in the sense that if we perform the same statements several times, we do not get the same time.

We solve this problem by performing the same computation several times and average the results.

We perform the same computation 10 times, and then display the minimum, the maximum and the average time.

timer

The *timer* function, measures the system time. This corresponds to the time required by the CPU to perform the specific computation required by Scilab, and not by other processes.

2.1 USER VS CPU TIME

In this section, we analyze the difference between the user and CPU time on multi-core systems.

The following session presents the result when we run on a Windows machine with 2 cores, where Scilab makes use of the multi-threaded Intel MKL.

```
-->disp ([ tUser tCpu tCpu / tUser ])
     0.5148033 1.0296066 2.0592132
```

We see that there is a significant difference between the CPU and the wall clock time. The ratio is close to 2, which is the number of cores.

Indeed, on a multi-core machine, if Scilab makes use of a multi-threaded library, the *timer* function reports the sum of the times spent on each core. This is why, we used the *tic* and *toc* functions to measure the performances of an algorithm.

3. SCILAB-5.3.0 VS SCILAB-5.3.0-SSE3

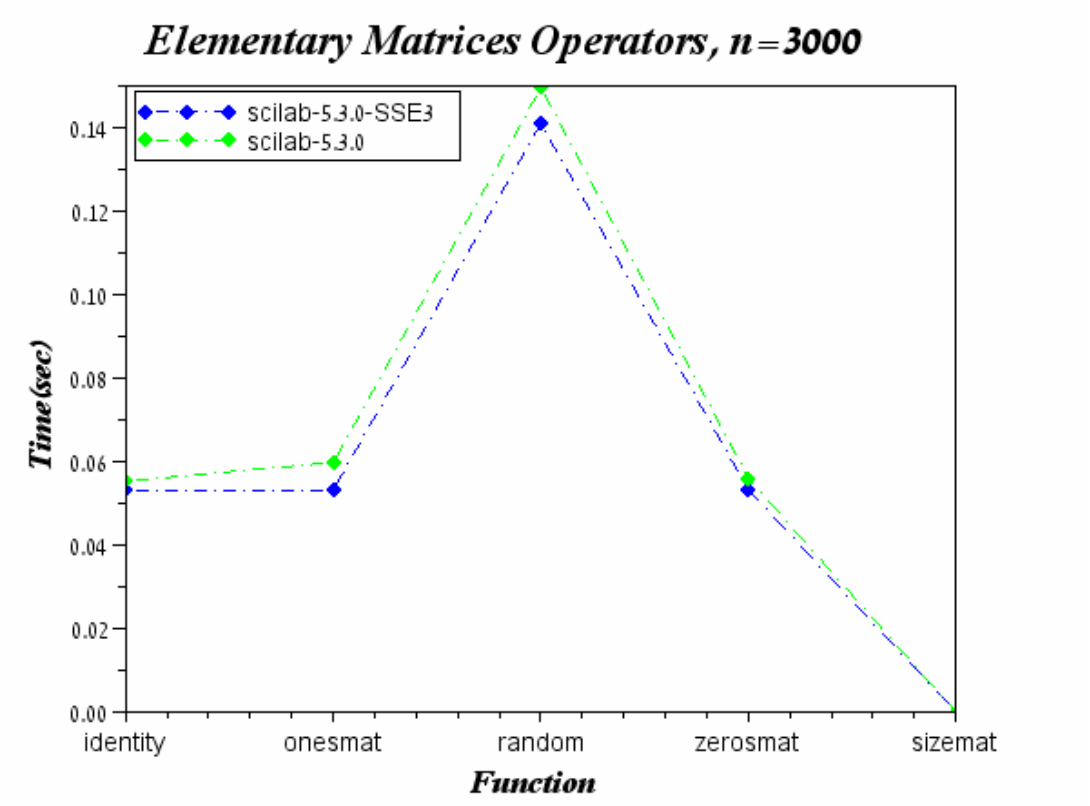
In this section, we present the outcome of Scilab versions on the performance of the general matrix-matrix product.

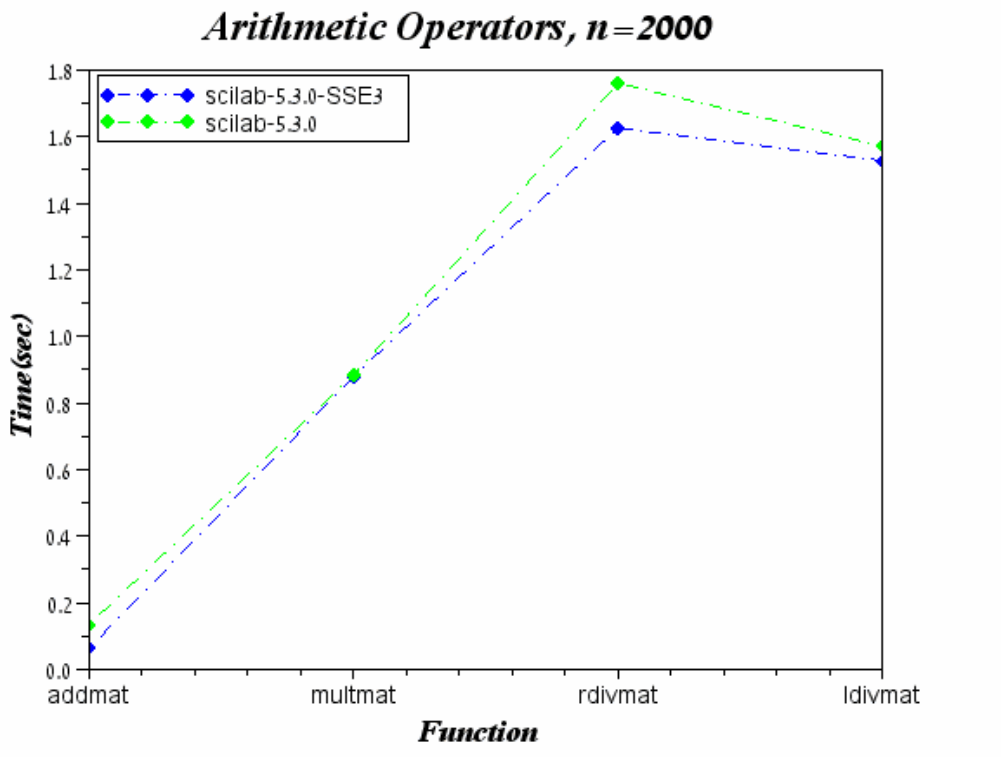
3.1 SCILAB MATH CORE FUNCTIONS

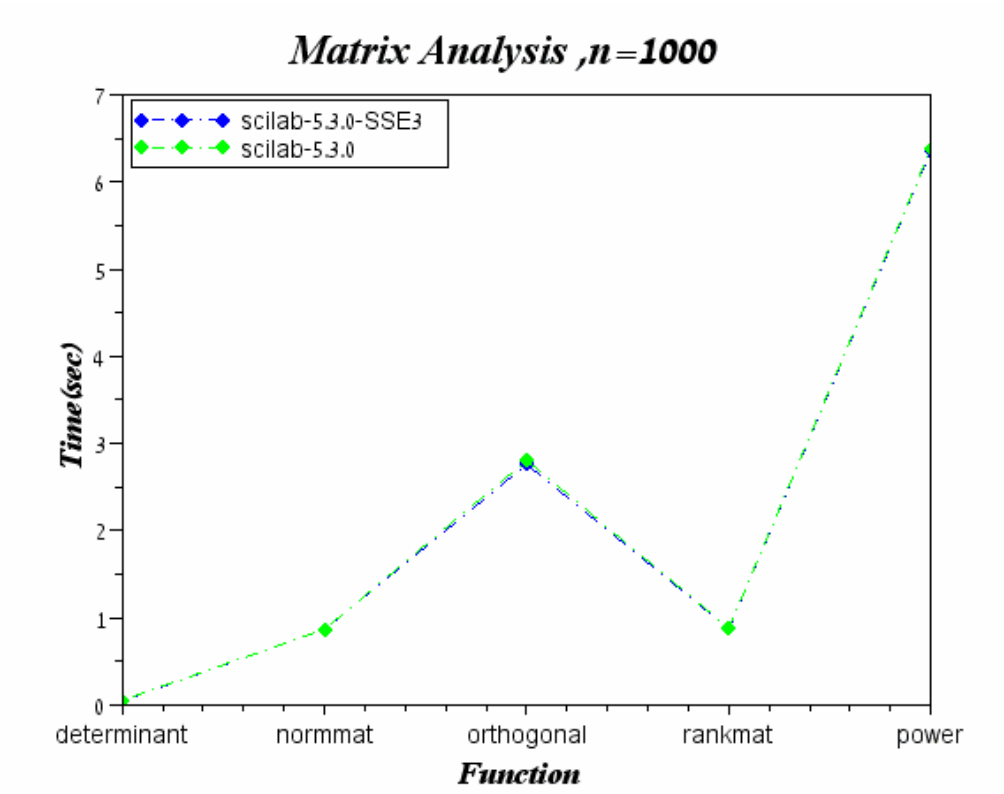
In this section, we compare the performance of Scilab math core operations.

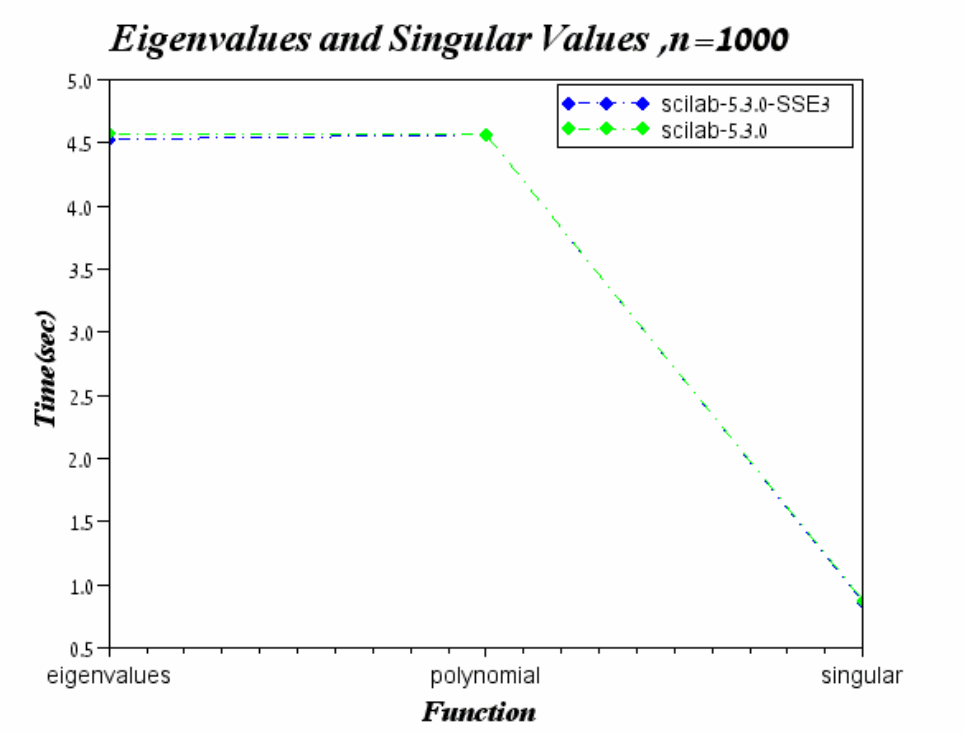
We repeat the timing 10 times to get a more reliable estimate of the performance.

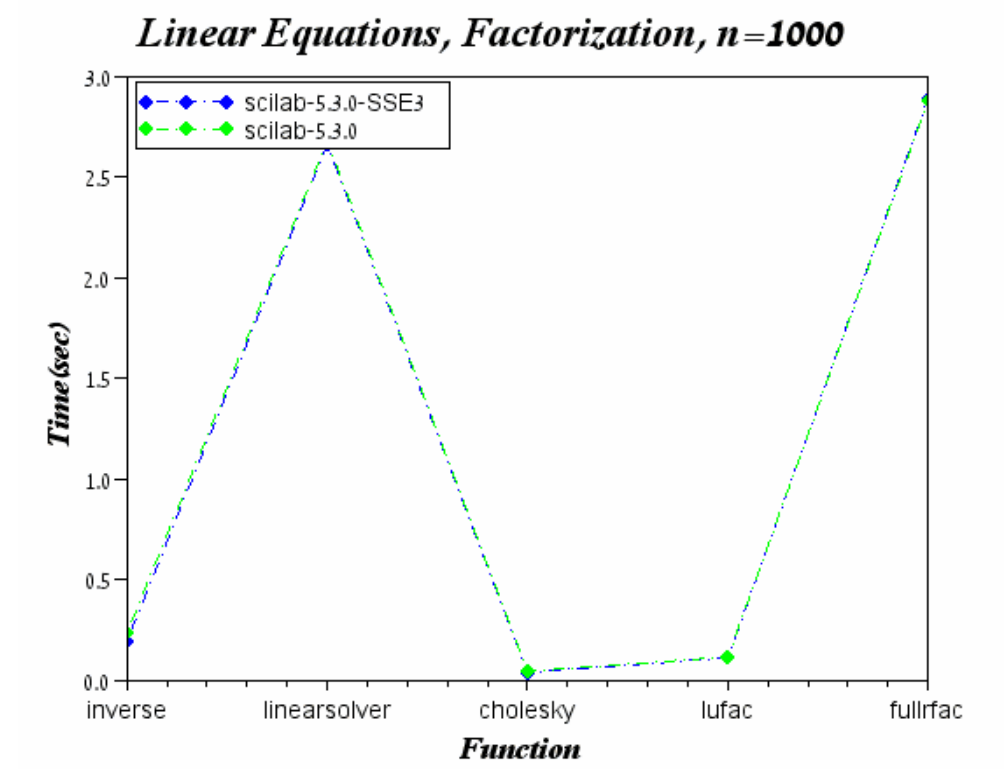
n – Represent the matrix order.

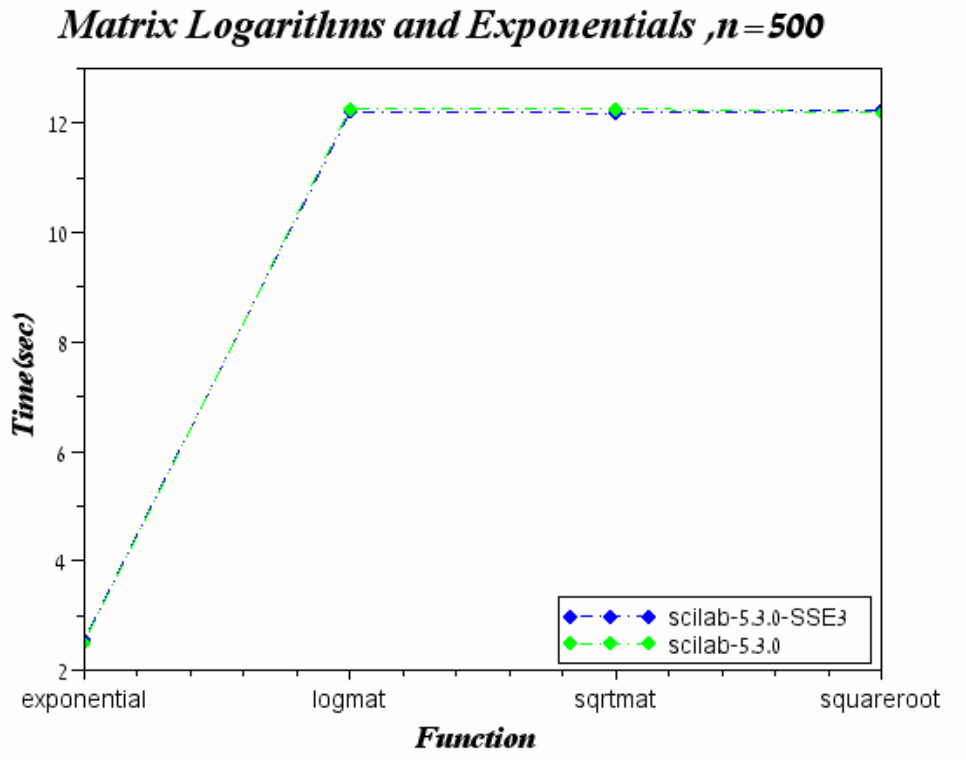


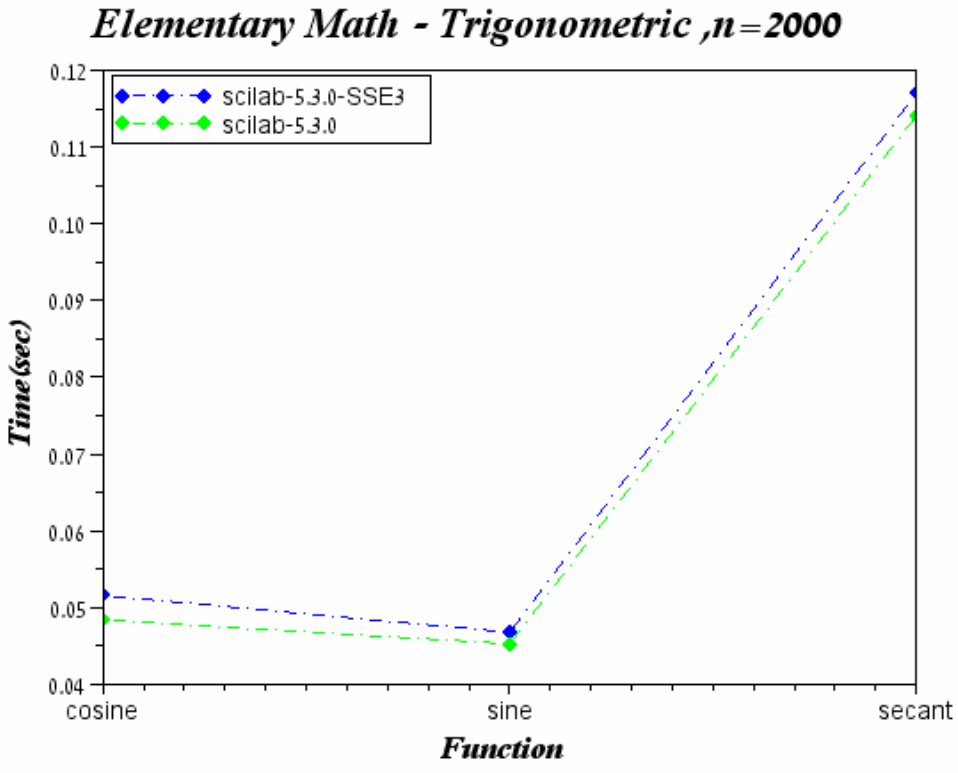

Arithmetic Operators, n=2000
scilab-5.3.0-SSE3
scilab-5.3.0
Time(sec)
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
addmat
multmat
rdivmat
ldivmat
Function


Matrix Analysis ,n=1000
scilab-5.3.0-SSE3
scilab-5.3.0
Time(sec)
7
6
5
4
3
2
1
0
determinant
normmat
orthogonal
rankmat
power
Function


Eigenvalues and Singular Values ,n=1000
scilab-5.3.0-SSE3
scilab-5.3.0
Time(sec)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
eigenvalues
polynomial
singular
Function


Linear Equations, Factorization, n=1000
scilab-5.3.0-SSE3
scilab-5.3.0
Time(sec)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
inverse
linearsolver
cholesky
lufac
fullrfac
Function

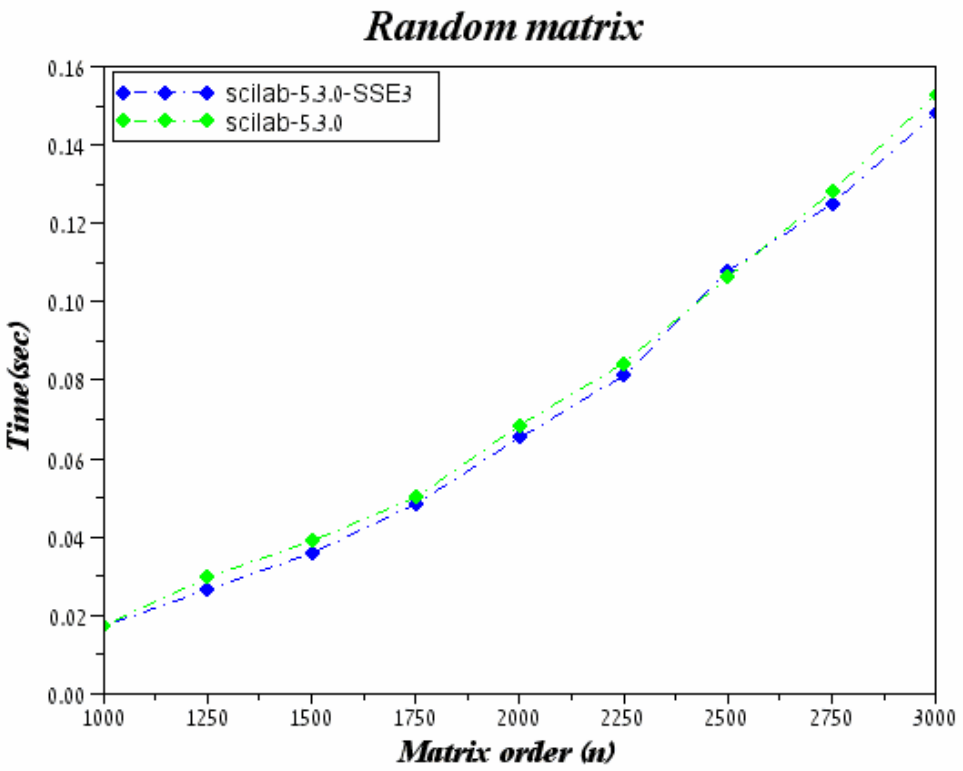

Matrix Logarithms and Exponentials ,n=500
Time(sec)
12
10
8
6
4
2
exponential
logmat
sqrtmat
squareroot
Function
scilab-5.3.0-SSE3
scilab-5.3.0

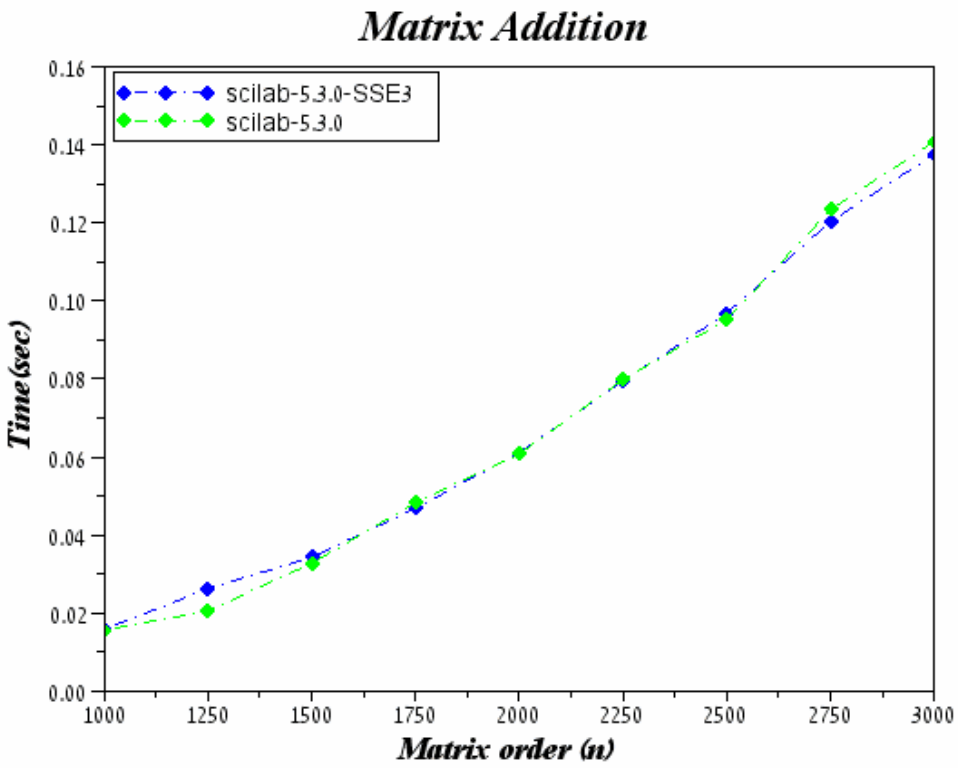

Elementary Math - Trigonometric ,n=2000
Time(sec)
0.12
0.11
0.10
0.09
0.08
0.07
0.06
0.05
0.04
cosine
sine
secant
Function
scilab-5.3.0-SSE3
scilab-5.3.0

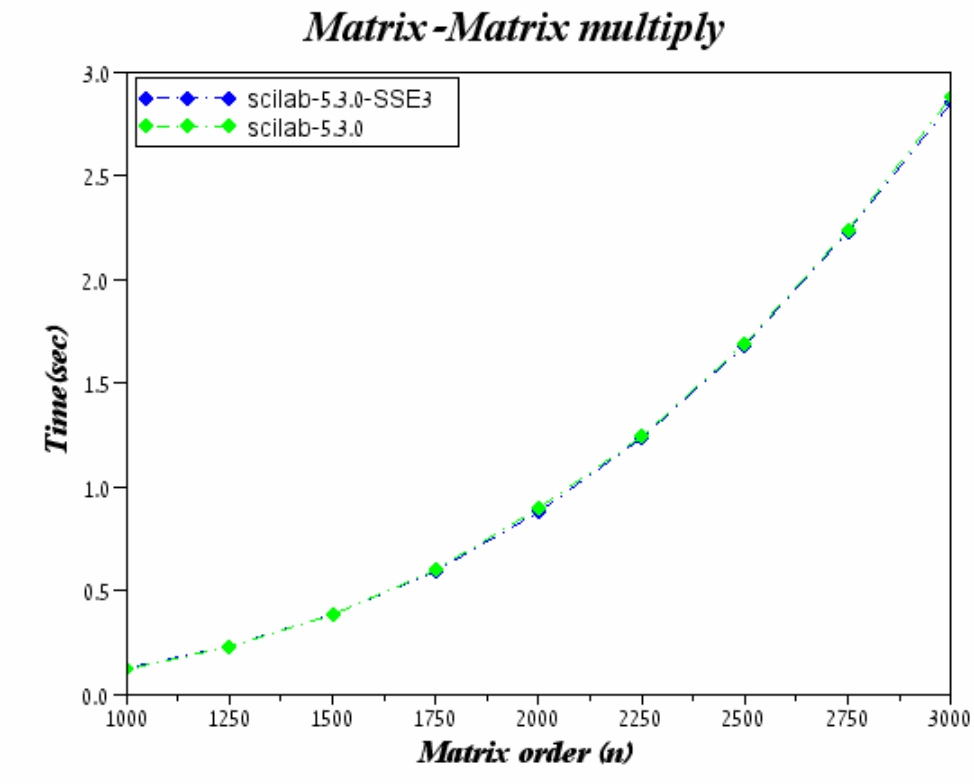
3.2 INCREASING MATRIX ORDER

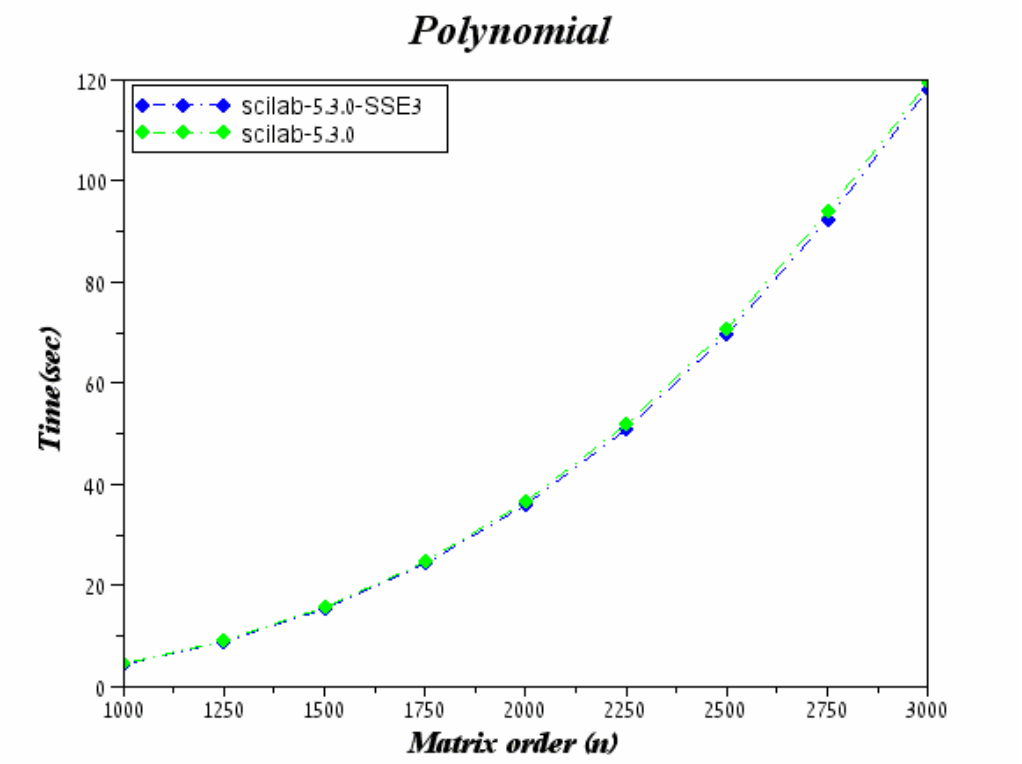
In this section, we compare the performance of Scilab math core operations with matrix order increasing.

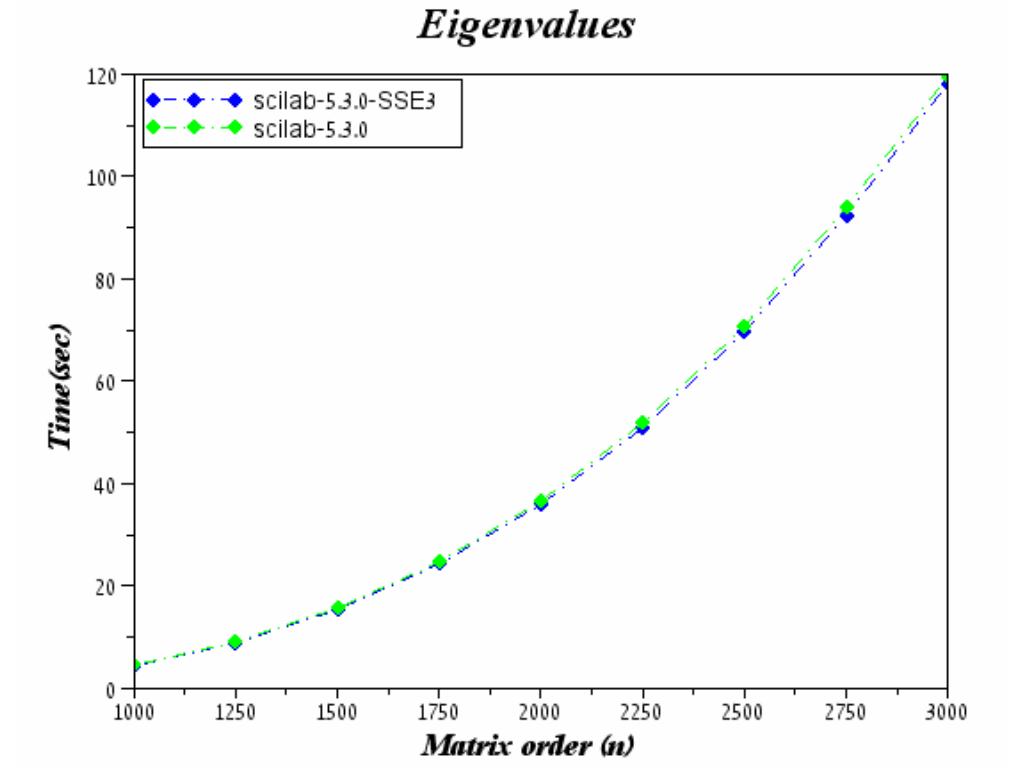
We repeat the timing 10 times to get a more reliable estimate of the performance.

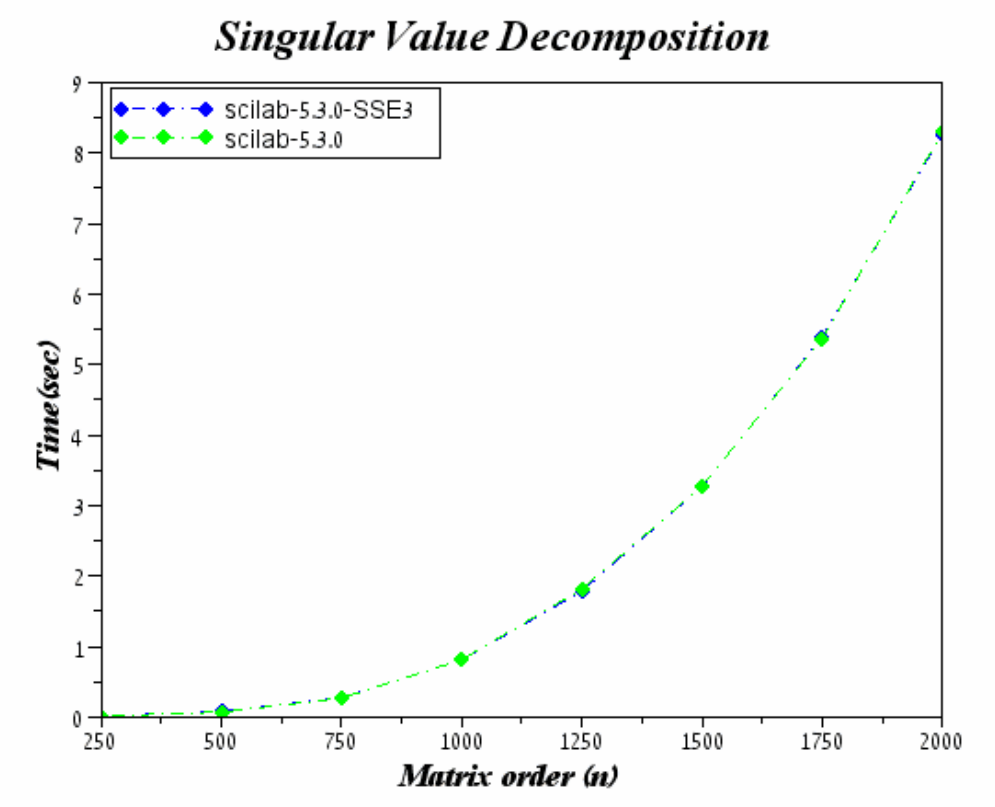



Matrix Addition
scilab-5.3.0-SSE3
scilab-5.3.0
Time (sec)
Matrix order (n)
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
1000
1250
1500
1750
2000
2250
2500
2750
3000


Matrix-Matrix multiply
scilab-5.3.0-SSE3
scilab-5.3.0
Time (sec)
Matrix order (n)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1000
1250
1500
1750
2000
2250
2500
2750
3000


Polynomial
scilab-5.3.0-SSE3
scilab-5.3.0
Time (sec)
Matrix order (n)
0
20
40
60
80
100
120
1000
1250
1500
1750
2000
2250
2500
2750
3000

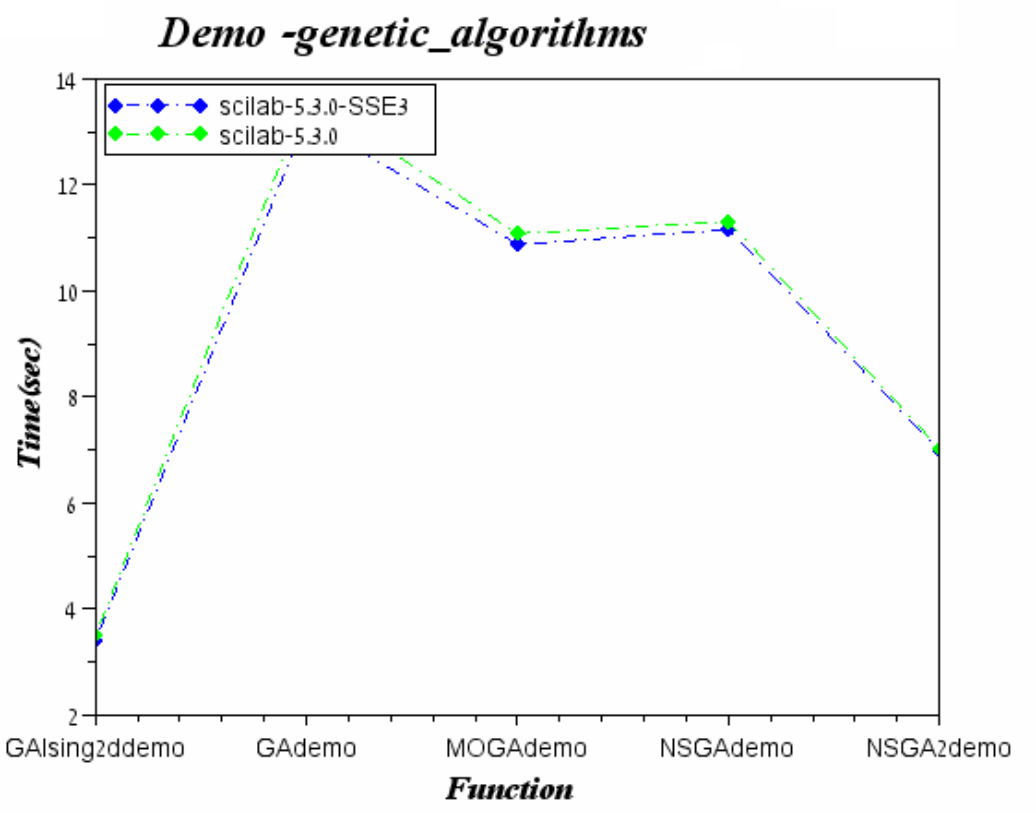

Eigenvalues
scilab-5.3.0-SSE3
scilab-5.3.0
Time (sec)
Matrix order (n)
0
20
40
60
80
100
120
1000
1250
1500
1750
2000
2250
2500
2750
3000

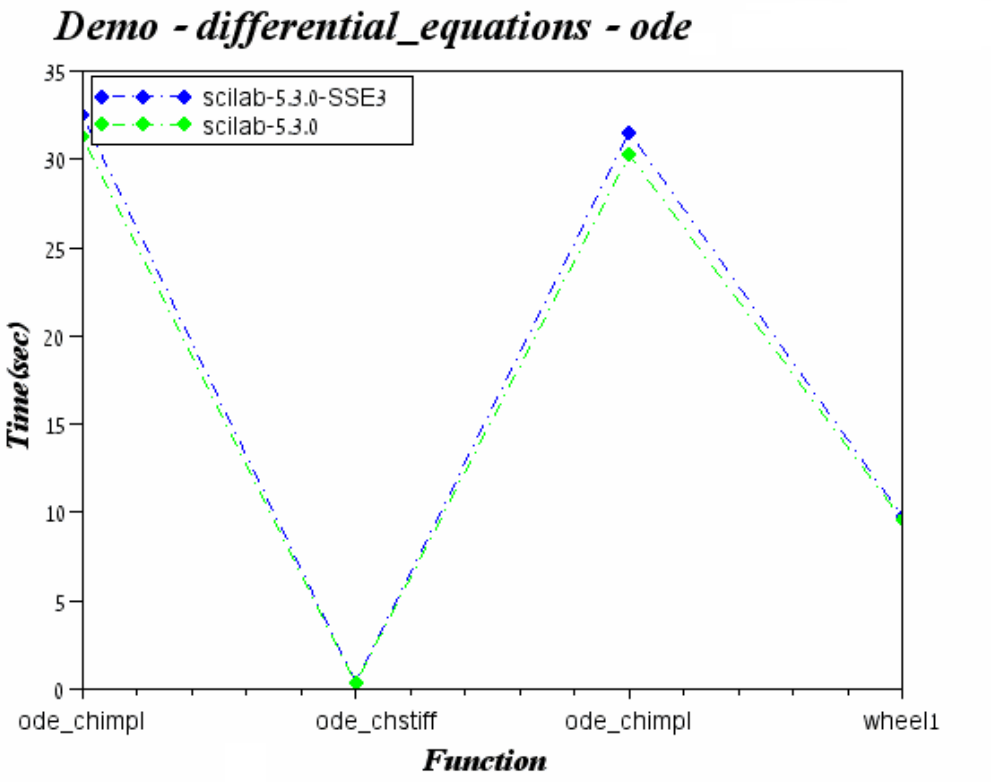

Singular Value Decomposition
scilab-5.3.0-SSE3
scilab-5.3.0
Time (sec)
Matrix order (n)
0
1
2
3
4
5
6
7
8
9
250
500
750
1000
1250
1500
1750
2000

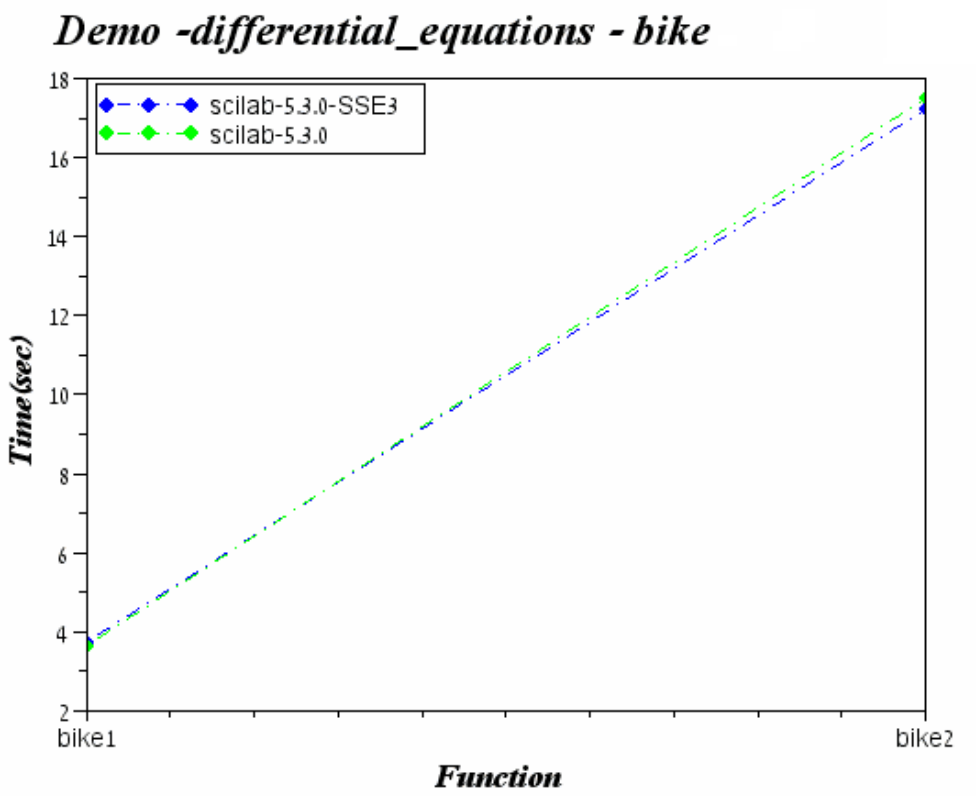
3.3 SCILAB DEMOS

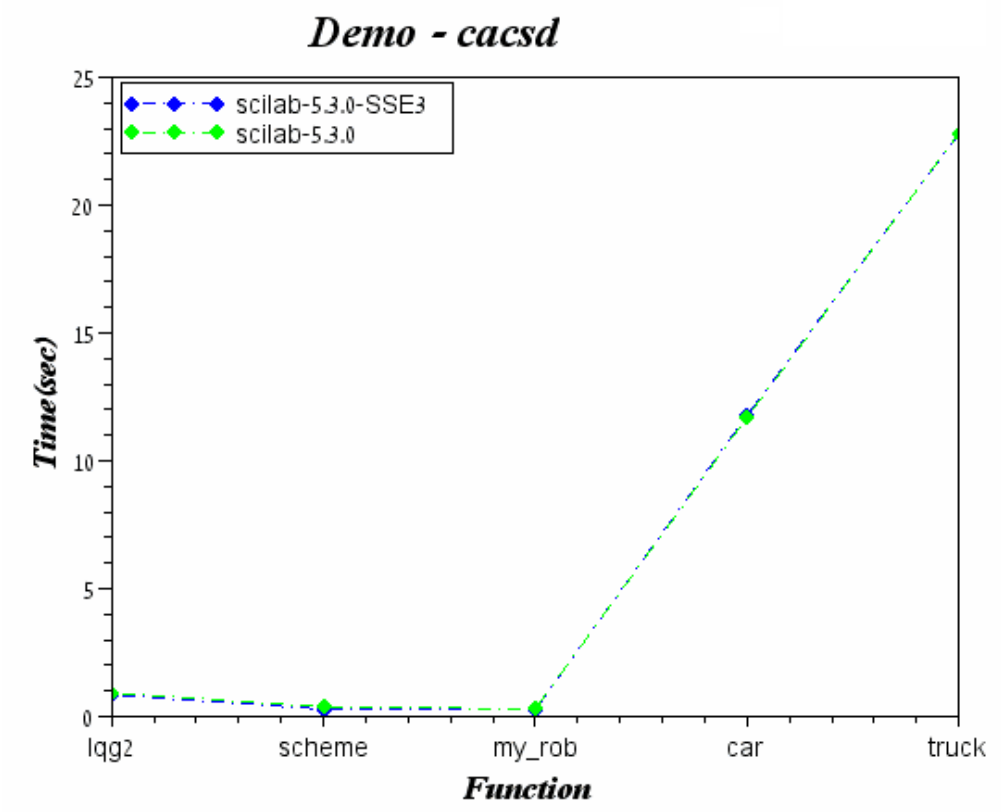
In this section, we compare the performance of Scilab Demos.

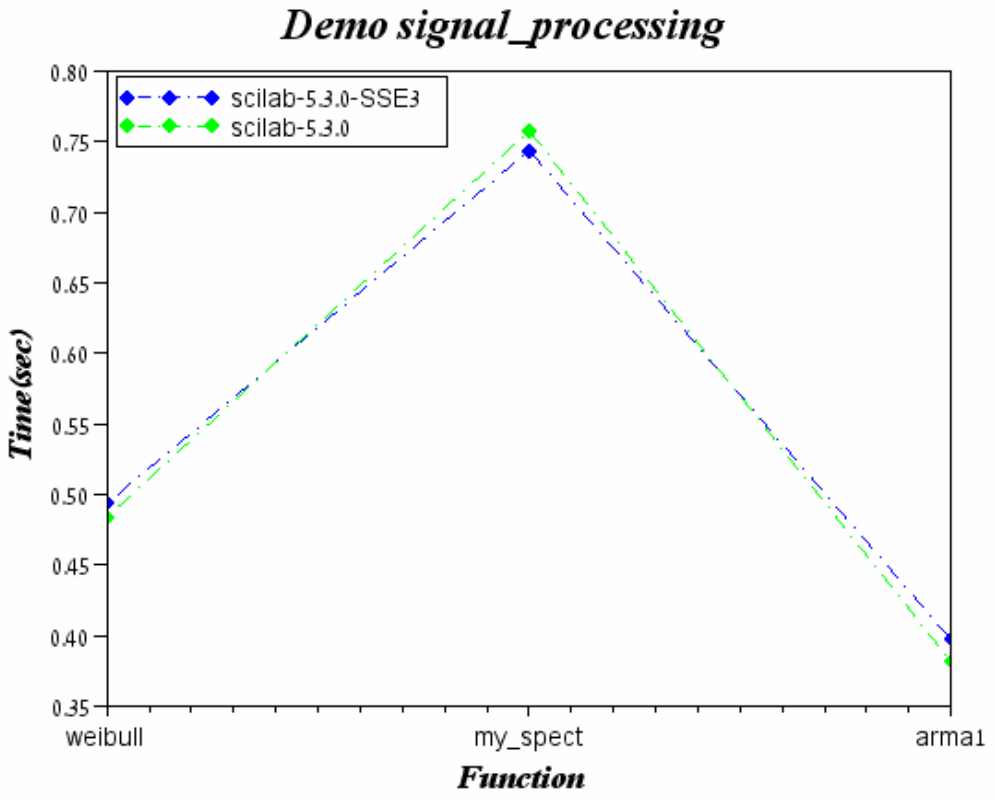
We repeat the timing 10 times to get a more reliable estimate of the performance.




Demo - differential_equations - ode
scilab-5.3.0-SSE3
scilab-5.3.0
Time (sec)
0
5
10
15
20
25
30
35
ode_chimpl
ode_chstiff
ode_chimpl
wheel1
Function


Demo -differential_equations - bike
scilab-5.3.0-SSE3
scilab-5.3.0
Time (sec)
2
4
6
8
10
12
14
16
18
bike1
bike2
Function


Demo - cacsd
scilab-5.3.0-SSE3
scilab-5.3.0
Time (sec)
0
5
10
15
20
25
lqg2
scheme
my_rob
car
truck
Function


Demo signal_processing
scilab-5.3.0-SSE3
scilab-5.3.0
Time (sec)
0.35
0.40
0.45
0.50
0.55
0.60
0.65
0.70
0.75
0.80
weibull
my_spect
arma1
Function

4. CONCLUSION

Benchmarks on operation based on MKL will have the same result between 5.3.0 and with 5.3.0-SSE3 because MKL uses all features of the CPU by default.

5. REFERENCES

[1] "Programming in Scilab", Michael Baudin, 2010. http://forge.scilab.org/index.php/p/docprogscilab/downloads/

[2] http://wiki.scilab.org/Linalg_performances