UNIVERSITY OF HELSINKI
DEPARTMENT OF PHYSICS

TOOLS FOR HIGH PERFORMANCE COMPUTING

Exercise 6

Student: Caike Crepaldi

Professor: Antti Kuronen

October 2015

Problem 1

In order to calculate the times, the original program source code had some small increments:

eigendouble.f90

```
12   real :: tf,ti
```

eigendouble.f90

```
39   ! ----- The eigenvalue calculation proper -----
40
41   call cpu_time(ti)
42   call dgeev('V','V',n,a,n,wr,wi,vl,n,vr,n,work,lwork,ok)
43   call cpu_time(tf)
44
45   write(6,*) tf-ti, n
```

After that, we can use the following bash script to do the compilations and print the CPU time into a file:

eigenscript.sh

```
1  #!/bin/bash
2
3  COUNTER=0
4  n=50
5  file1='time_eigenO3.data'
6  file2='time_eigenO0.data'
7  OPTIMIZATION=-O3
8
9  rm -f $file1 $file2
10
11 cd ~/Helsinki/THPC/crepaldi_caike_ex06/ex6p1/lapackf90/src
12 make cleanall
13 make FFLAGS=-O3
```

```
14 echo "lapack library: Compilation ($OPTIMIZATION)---> DONE"
15 cd ..
16 cd ..
17 gfortran eigendouble.f90 -llapack -Llapackf90/src
18 echo "eigendouble.f90: Compilation ---> DONE"
19
20 echo "File: $file1"
21 #echo "Optimization = $OPTIMIZATION"
22
23 while [ $n -le 2000 ]; do
24     ./a.out $n 8540585 >> $file1
25     echo "Iteration $COUNTER ---> DONE (n=$n)"
26     let COUNTER=COUNTER+1
27     let n=n+50
28 done
29
30 COUNTER=0
31 n=50
32 OPTIMIZATION=-O0
33
34 cd ~/Helsinki/THPC/crepaldi_caike_ex06/ex6p1/lapackf90/src
35 make cleanall
36 make FFLAGS=-O0
37 echo "lapack library: Compilation ($OPTIMIZATION)---> DONE"
38 cd ..
39 cd ..
40 gfortran eigendouble.f90 -llapack -Llapackf90/src
41 echo "eigendouble.f90: Compilation ---> DONE"
42
43 echo "File: $file2"
44 #echo "Optimization = $OPTIMIZATION"
45
46 while [ $n -le 2000 ]; do
47     ./a.out $n 8540585 >> $file2
48     echo "Iteration $COUNTER ---> DONE (n=$n)"
49     let COUNTER=COUNTER+1
50     let n=n+50
51 done
```

Note. In order to use the script, please change the **cd** commands using the path to your own directory.

After running the script, two new archives shall appear in your current directory:

"*time_eigenO3.data*" and "*time_eigenO0.data*".

The content of my version of this two files can be seen in the appendix.

By analysing the graphics in the figures 1, 2, 3, 4, we can conclude that the calculation have a scalability of cubic order (compare the residuals of the cubic and quadratic versions). Therefore, we can say that the scaling, in the big-O notation, is $\mathbf{O}(\mathbf{n^3})$. This result is compatible with what was expected by the literature[1].

[1] Source: `http://www.netlib.org/lapack/lug/node70.html`. Credits to Andrey for helping me find this out.

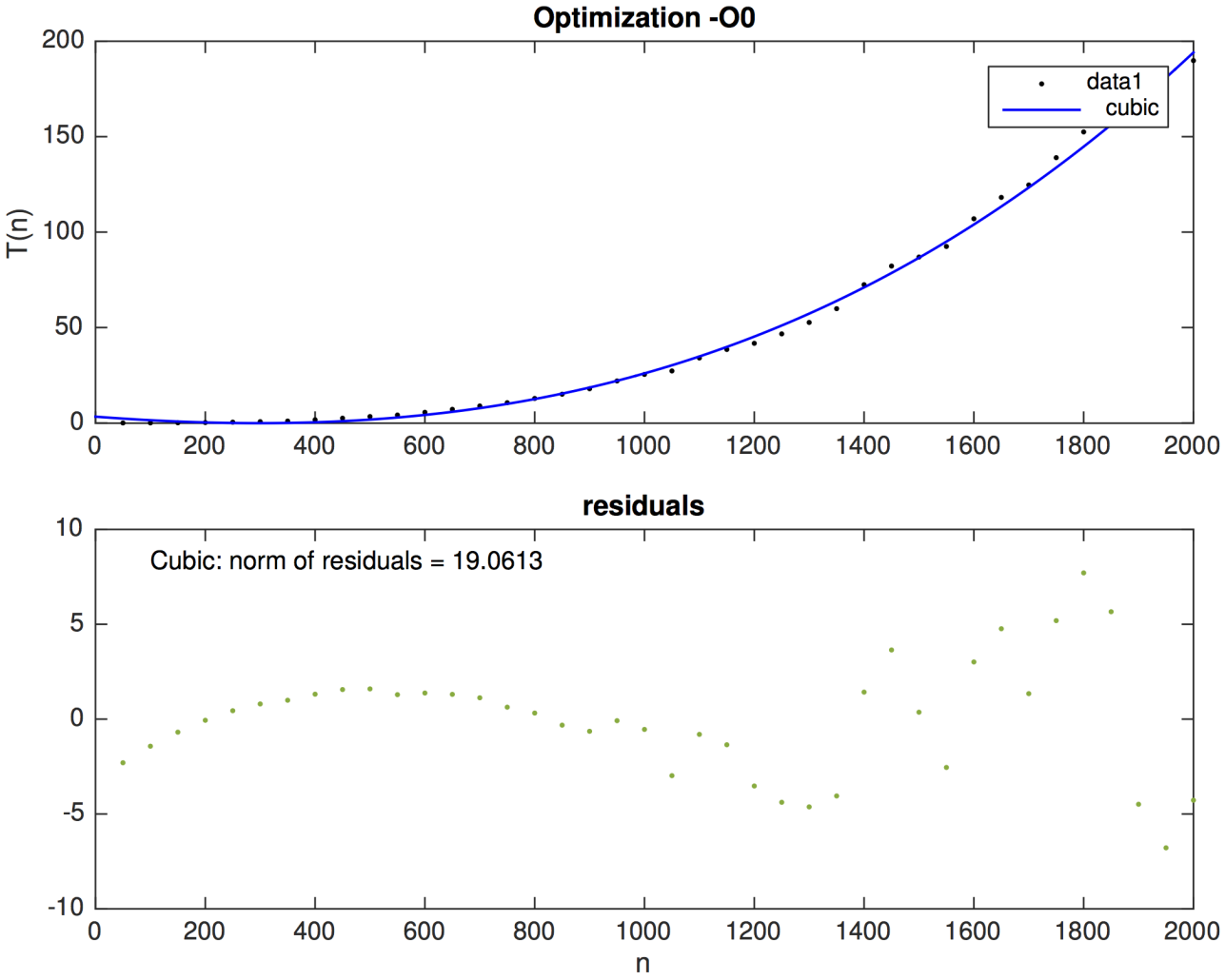

Figure 1: -O0 option with cubic fit.

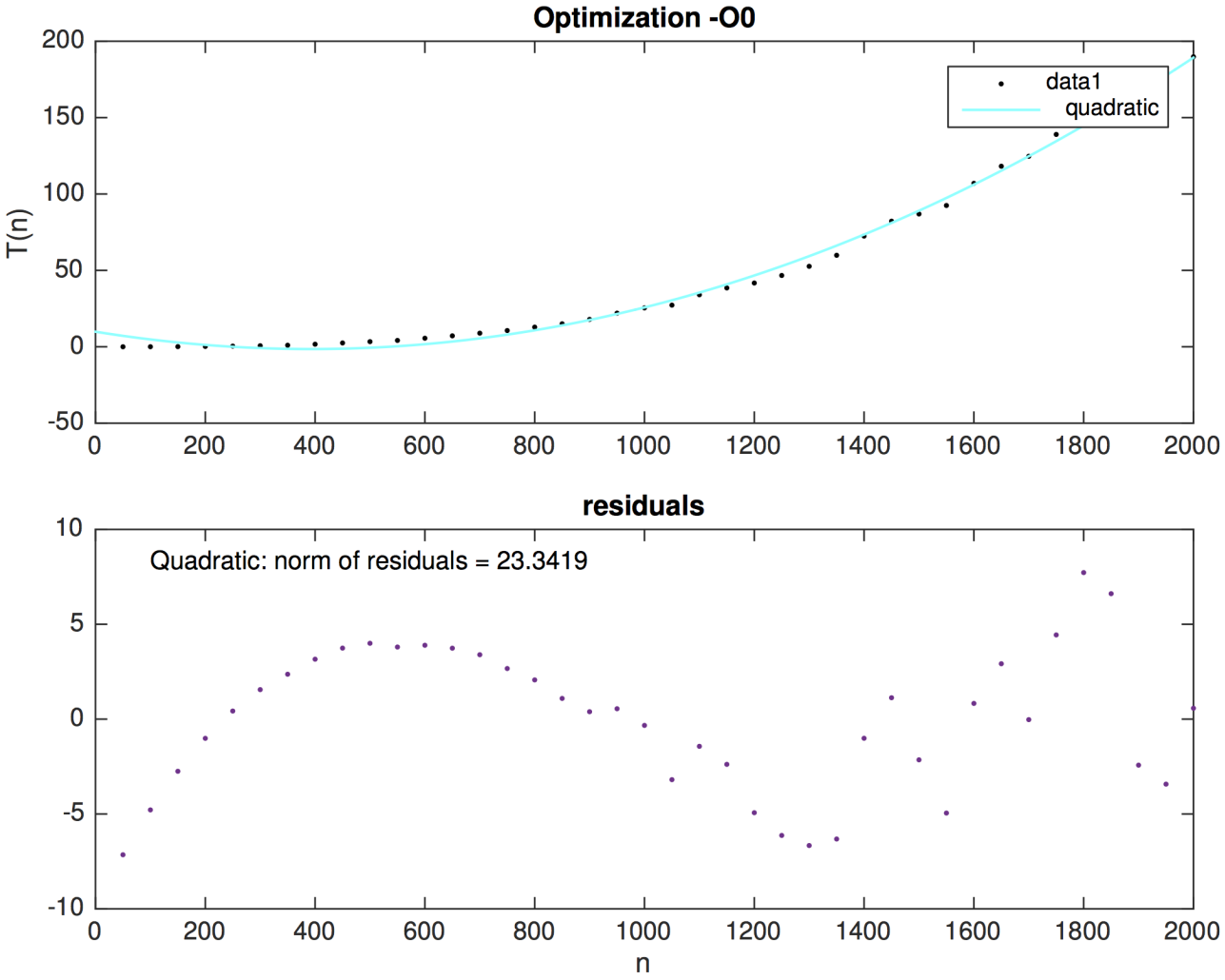

Figure 2: -O0 option with quadratic fit.

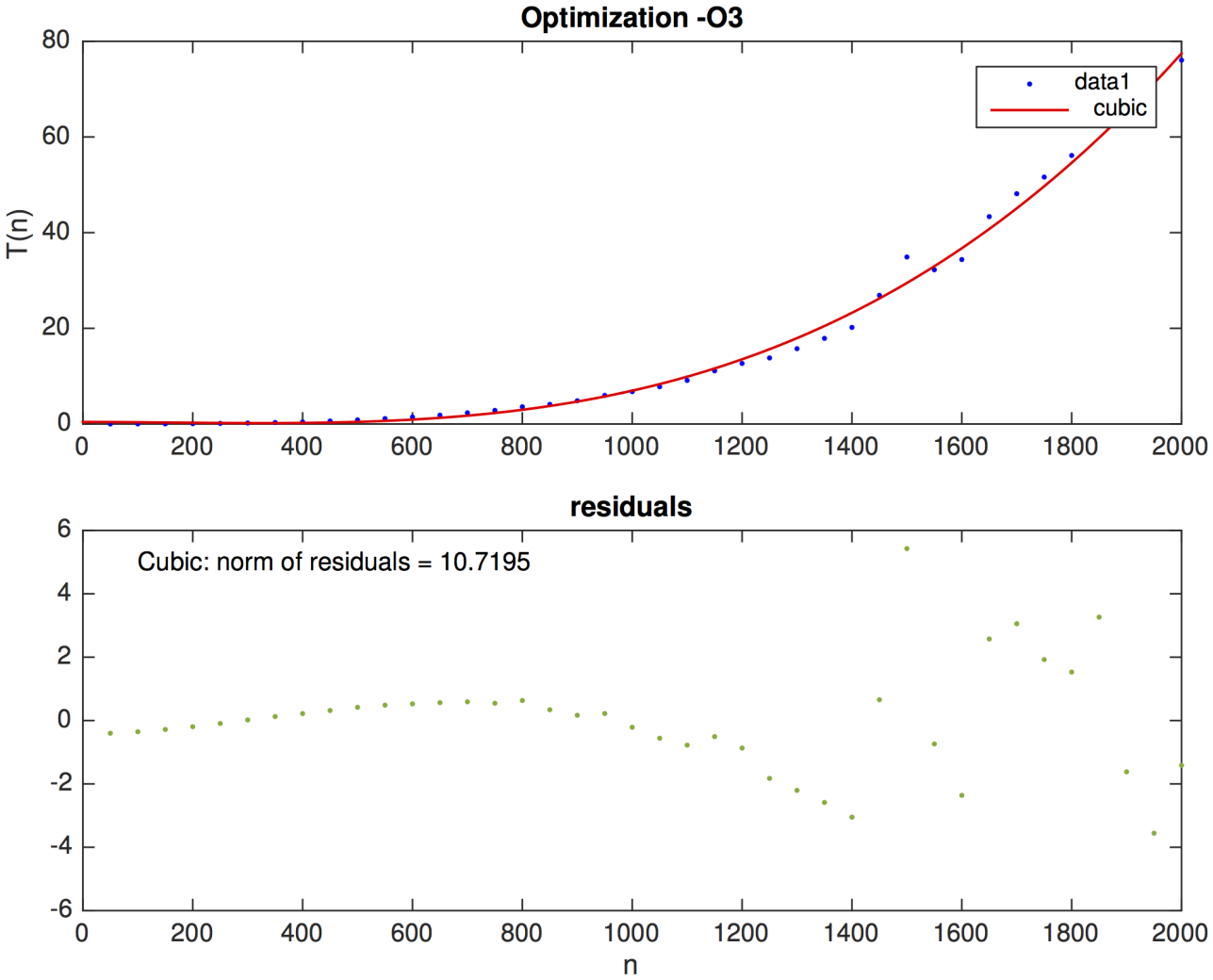

Figure 3: -O3 option with cubic fit.

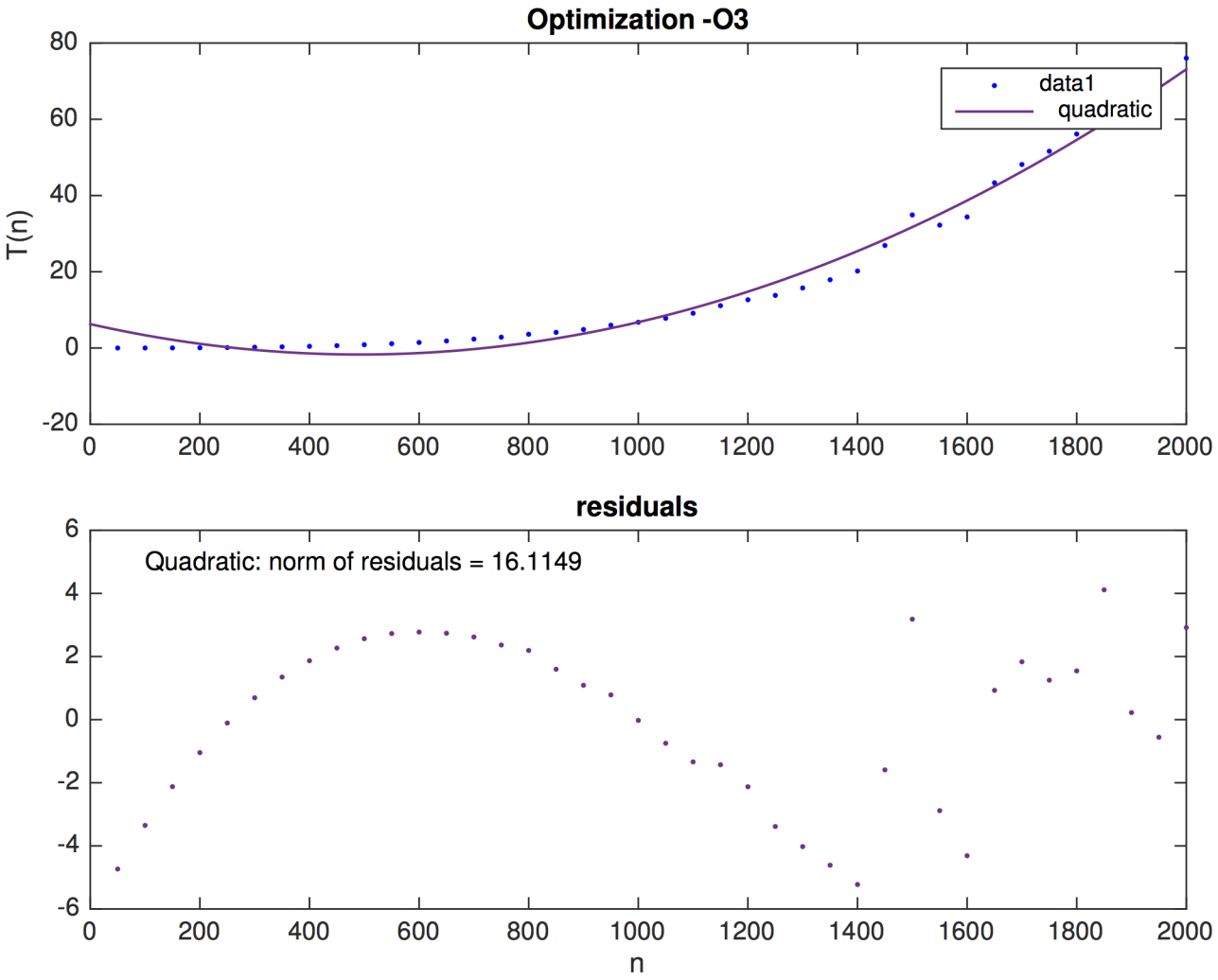

Figure 4: -O3 option with quadratic fit.

Problem 2

In order to do such task, I created the bash script below:

maximumscript.sh

```
#!/bin/bash

FILE=$1
COUNTER=0

rm -f times.data

while read line; do
    let COUNTER=COUNTER+1
    done < $FILE
WL=0

while read line; do
    let WL=WL+1
    FFLAGS=$line
    gfortran $FFLAGS -o prog maximum.f90
    ./prog >> times.data
    echo "command $FFLAGS done: ($WL/$COUNTER)"
done < $FILE
```

To use the script, use the following command in the shell environment:

Command 1

```
$ ./maximumscript.sh gcc_options.txt
```

Then, a file named times.data will be created with the output of the maximum program compiled with the several FFLAGS listed in the gcc_options.txt file.

The (plotted) result can be seen in figure 5.

Lowest CPU times (in order):

1. -O2
2. -O
3. -O1
4. -Os
5. -O3
6. -Ofast
7. -Og
8. -ffast-math
9. -fshrink-wrap
10. -fno-signed-zeros

Highest CPU times (in order):

1. -ftree-ccp
2. -ftree-builtin-call-dce
3. -fvect-cost-model
4. -fstrict-aliasing
5. -fweb
6. -ftree-switch-conversion
7. -fwhole-program
8. -ftree-loop-distribution
9. -ftree-coalesce-inline-vars
10. -fstrict-overflow

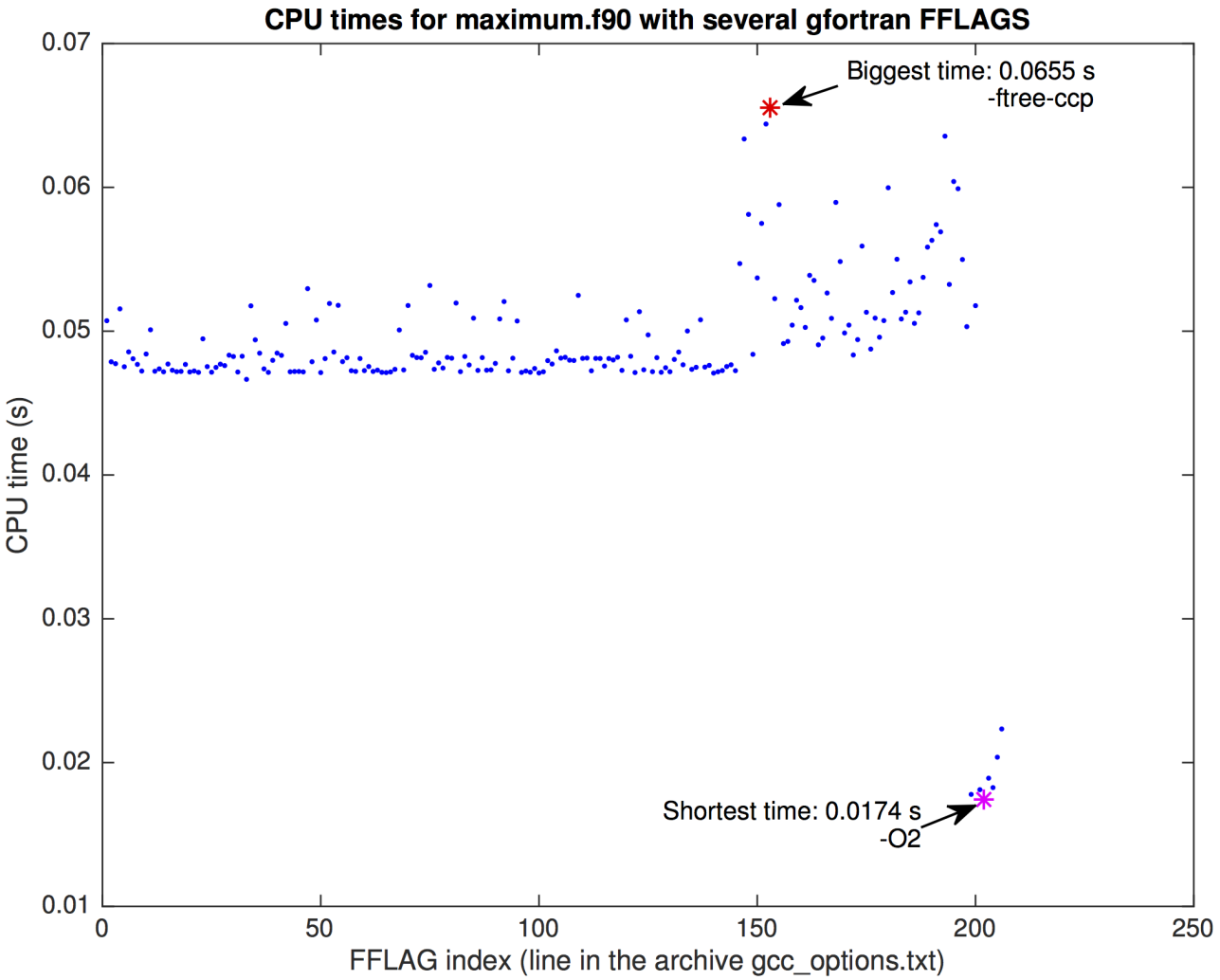

Figure 5: Biggest and shortest CPU times in comparision with the rest of the data.

Problem 3

(a) MPI, a.k.a. Message Passing Interface, is a standardized and portable message-passing system that defines the syntax and semantics of a core of library routines useful for writing portable message-passing programs in different computer programming languages (Fortran, C, C++ and Java). In other words, it is a collection of routines for facilitating communication among the processors in a distributed-memory parallel program. A low point of this system would be that it is intended to be used with Fortran 77 and ANSI C, and therefore doesn't support modern Fortran advanced features. Since message passing is a low level parallelization method, the programmer has to do everything himself, but this is the only way to write efficient and scalable parallel code for more than ten processors. Some efficient implementations of MPI are free or in the public domain, such as the Open MPI implementation. MPI offers portability, standardization, performance, functionality, and several high quality implementations.

OpenMP, a.k.a. Open Multi-Processing, is an application programming interface that supports multi-platform shared memory multiprocessing programming in C/C++, and Fortran, on most platforms, processor architectures and operating systems and consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior. It uses a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications for platforms ranging from the standard desktop computer to the supercomputer. OpenMP has been implemented in many commercial and GNU compilers, e.g. the latest version (OpenMP 4.0) is supported by GNU GCC (C/C++) 4.9.0, GNU GCC (Fortran) 4.9.1, and Intel Fortran and C/C++ compilers 15.0.

(b) See figure 6. The source is `http://www.top500.org`. The information about the number of cores and etc. comes from their excel table that can be downloaded in their website.

Name	Site	Manufacturer	Country	Segment	Total Cores
Tianhe-2 (MilkyWay-2)	National Super Computer Center in Guangzhou	NUDT	China	Research	3120000
Sequoia	DOE/NNSA/LLNL	IBM	United States	Research	1572864
Shoubu	Institute of Physical and Chemical Research (RIKE	PEZY Computing / Exascaler	Japan	Research	787968
Mira	DOE/SC/Argonne National Laboratory	IBM	United States	Research	786432
	RIKEN Advanced Institute for Computational Scien	Fujitsu	Japan	Research	705024
Titan	DOE/SC/Oak Ridge National Laboratory	Cray Inc.	United States	Research	560640
Stampede	Texas Advanced Computing Center/Univ. of Texas	Dell	United States	Academic	462462
JUQUEEN	Forschungszentrum Juelich (FZJ)	IBM	Germany	Research	458752
Vulcan	DOE/NNSA/LLNL	IBM	United States	Research	393216
C01N	Tulip Trading	Supermicro/SGI	Australia	Industry	265440
Suiren Blue	High Energy Accelerator Research Organization /K	PEZY Computing / Exascaler	Japan	Research	263168
Suiren	High Energy Accelerator Research Organization /K	PEZY Computing / Exascaler	Japan	Research	262784
	Government	Cray Inc.	United States	Governme	225984
QUARTETTO	Research Institute for Information Technology, Kyu	Hitachi/Fujitsu	Japan	Academic	222072
Shaheen II	King Abdullah University of Science and Technolog	Cray Inc.	Saudi Arabia	Academic	196608
cascade	DOE/SC/Pacific Northwest National Laboratory	Atipa Technology	United States	Research	194616
Tianhe-1A	National Supercomputing Center in Tianjin	NUDT	China	Research	186368
Pleiades	NASA/Ames Research Center/NAS	SGI	United States	Research	185344
Tianhe-2 LvLiang Soluti	LvLiang Cloud Computing Center	NUDT	China	Industry	174720
Fermi	CINECA	IBM	Italy	Academic	163840
Hopper	DOE/SC/LBNL/NERSC	Cray Inc.	United States	Research	153408
Garnet	ERDC DSRC	Cray Inc.	United States	Research	150528
SuperMUC	Leibniz Rechenzentrum	IBM/Lenovo	Germany	Academic	147456
Abel	Petroleum Geo-Services	Cray Inc.	United States	Industry	145920
Cielo	DOE/NNSA/LANL/SNL	Cray Inc.	United States	Research	142272
Tera-100	Commissariat a l'Energie Atomique (CEA)	Bull, Atos Group	France	Research	138368

Figure 6: The largest supercomputers nowadays.