A set of simple benchmark programs
for representative cases
# in X-ray astronomy

prepared by L.Chiappetti - IFCTR

17 Aug 1990
revised 23 Aug 1991
revised 28 Dec 1992

1. Introduction

I have prepared a set of simple programs which can be used as benchmarks of the performance of different CPUs in cases which are representative of the usual workload for typical X-ray astronomical analysis.

There are two main benchmark programs and an ancillary one. The two main programs (which are essentially benchmarks of CPU performance) are a spectral fitting program and a direct Fourier transform program. The ancillary program is used to generate the matrix needed by the fitting program, and can be seen as a limited i/o performance benchmark.

Please note that I have **willingly** excluded for the present tests on one hand, among the CPU-intensive tasks, any image processing program (as this is not prevailing in, or typical of, X-ray astronomy only), and, on the other hand any explicit test of i/o performance (in fact i/o performance is often more conditioned by the way it is coded than by the intrinsic performance of the computer; I mean that on the same machine, coding in plain Fortran, or in machine-dependent Fortran, or calling system-routines at various level can drastically affect i/o speed).

I have placed some care in making the benchmark programs (and the associated data) portable, so that they can easily be tested on other machines (at the only expense of writing a simple, machine-dependent, routine to measure time, whose interface is defined very clearly.

The source of the benchmark programs is available to anybody for further testing on other types of machines than those considered here. In fact I would encourage the readers of this note to obtain the source code, and test it on other models of Vaxes and workstations (DecStations, HP ...).

Further tests have been done one year after the first issue of the document on other models of Unix workstations (Sun and DECstation). The sections added during this revision in the document are outlined by a vertical bar on the right margin.

Further tests were added later on Sun IPX and using Sun Fortran 1.4 compiler. The sections modified at such time are outlined by a double bar

2. What kind of programs ?

As I said above there are two main programs and an ancillary one. Each of the three is self-contained in a single source file, so that it can be compiled and linked in a strightforward way, after some minimal editing (described in section 3.2). The timing program is a plain CPU test and has no data associated with it, while the fitting program has two data files associated with it. Both are distributed as plain ASCII files, but the ancillary program has to be used on one of those to convert it to the binary form used by the actual fitting module. The format of the distribution is described in section 3.1.

Both programs run only on a fixed dataset, as the purpose of the test is not to assess the performance of the program with different datasets, but of one program with the same dataset under different conditions (machines, compilers, or compiler options e.g. optimization level).

It has to be noted that both programs have been derived by REAL programs (used in Milano as part of our Exosat system), and therefore are representative of real conditions in which these algorithm are likely to be used.

2.1 The direct Fourier transform

This program performs a date-compensated direct Fourier transform (DCDTF) according to the method of Ferraz-Mello (1981, A.J. **86**,619), which is slightly more complicated then the usual (Deeming or Scargle) DFTs. For portability, as the performance does not depend on the content of the data, a time series (pure sinusoid) is generated internally.

The user can specify the number of points in the time series, and the number of frequency steps for the DCDTF, up to a maximum which is encoded in the program at compile time (see 3.2). The extrema of the frequency range are selected by the program.

This benchmarks tests essentially CPU-performance, and measures two time intervals (carefully excluding the few terminal i/o) :

the time for generation of the time series
the time for the computation of the periodogram

2.2 The fitting program

This program performs a CURFIT-fitting of a representative X-ray spectrum. The input spectrum and the response matrix are fixed, and are supplied as external files. The spectrum is real astronomical data (an Exosat spectrum of Cyg X-2, including LE filter data, ME Argon and Xenon data), and is fitted using a two component model (blackbody plus Bremsstrahlung with interstellar absorption). The initial guesses are encoded in the program, and are chosen slightly off the best fit value to increase the number of iterations (hence the time of the fit, and implicitly the precision in the measurement of the elapsed time), but not so off to be prone to unpredictable numeric effects (which may be different machine to machine).

The particular dataset has been chosen as it is an example of a good statistics, moderately complex spectrum. Also the convolution is made quite unusual by the inclusion of three instruments.

The program has been simplified with respect to the original code, removing all references to unused spectral forms (however the sequence of subroutine calls and the complex arrangement of COMMON blocks are virtually unchanged). Also all dialogue concerning the choice of the fitting parameters has been removed, as well as all terminal and disk output generation at the end. I have also incorporated in the source file all references to external INCLUDE files, as these are inherently non- portable.

However some terminal output has been left in (namely the display of the various iterations), not only in order to allow checking that the program is performing as expected, but because this is actually the situation in **REAL LIFE**, where one is interested in the compound performance of the number-crunching and the display of results.

This program therefore does not therefore test plain CPU performance (computations), but a combination of that, of the overhead of subroutine calling (which in programs like this, contrary to the DFT case which can be encoded in a single routine, is very important), and, marginally, of terminal and i/o performance.

The benchmark measures the following time intervals:

the time to read in the spectrum from disk
the time to read in the matrix from disk
the time for preliminaries (initial guess, etc.); marginal
the time for the fitting loop (inclusive of display)

2.3 the ancillary program

The fitting program requires a spectrum and a response matrix. In order not to encumber the program with the book-keeping required by a real-life system, these are NOT in the original format used in the Exosat system in Milano (in fact there is not even the slight correspondence) but in a simplified format (deprived of all header information unnecessary here, and modified for portability), namely :

the spectrum is kept in a plain ASCII table
the matrix is kept in a binary file (n records of m floating point elements each).

While the choice of an ASCII format for the spectrum, even if it may not necessarily correspond to the real case, has little effect on the overall time, the choice of a binary format for the matrix is mandatory to ensure similarity with real life conditions.

However a binary floating-point file is intrinsically **unportable**, therefore it has been necessary to devise a mean for the export and exchange of the matrix data among the various machines (suitable e.g for ethero geneous networks). This is almost compulsory to be an ASCII formatted dump, with record length .le. 80 bytes. However, considering the large dynamic range, and in order not to lose precision at either ends, the data has to be written with many significant figures (e.g. a format E12.6). This, associated with the big size of the matrix (416*49), can easily make the dataset too large for the weakest network connections (Decnet).

Therefore a further reduction has been applied, removing those parts of the matrix which have only zero-content (remember the matrix is the juxtaposition of three separate matrices ...), which however needs some additional book-keeping. This way the file is only 1107 records instead of 3430.

Hence the existence of an ancillary program, which could be used only once before running the fitting program, to re-create the binary matrix file from the ASCII distribution file.

As a side effect, this program could be repeatedly run, under different conditions, to test i/o performance. The program actually measures the following times :

time to read in memory the matrix from the ASCII distribution file
time to output to disk in binary form the matrix itself.

3. The distribution

I can supply via e-mail or IBM-PC floppy disk the files described below to anybody who is interested in repeating the benchmarks on other machines. Please make sure you have enough spool space, disk quota or whatever space to receive them (particularly the ASCII matrix). Depending on the status of my and your connections, it is possible that I be able to supply the binary file directly (to be arranged case by case).

3.1 Format of the distribution files

There are three program source files and two data files in the distribution. The third data file (binary matrix) is not in the distribution and has to be recreated at each local site running program BENCHMAT. The files are :

```
source for BENCHDFT ( 275 lines)                22 kbyte
source for BENCHFIT (1793 lines)               141 kbyte
source for BENCHMAT ( 212 lines)                18 kbyte

ASCII spectrum (48 records)                      3 kbyte
ASCII matrix (intermediate)                     87 kbyte

binary matrix (final)                           80 kbyte
```

There are additional files for some systems (e.g. EXECs for IBM), but these are not considered to be of general use and are not included in the distribution.

3.2 How to use the distribution files

The above files shall be assigned the appropriate names. These are the customary Fortran extension (FORTRAN, .FOR, .f etc.) for the sources of BENCHDFT, BENCHMAT and BENCHFIT, while for the data files the names and types are :

```
IBM:        SPECTRUM TEST,    MATRIX ASCII and MATRIX TEST
HP:         SPECTR,           MATASC,  ....and MATRIX
Vax:        SPECTRUM.TEST,    MATRIX.ASCII and MATRIX.TEST
Unix:       spectrum.test,    matrix.ascii and matrix.test
MSDOS:      SPECTRUM.TST,     MATRIX.ASC ..and MATRIX.TST
```

step a: provide a time measuring routine

The **first operation** you shall do is to write a routine which measures elapsed or CPU time (or both) for your system, according to the specifications below.

Such a routine is already available for IBM, HP, VAX and BSD-like Unix (Sun and DEC Ultrix), and MS-DOS under IBM Professional Fortran. I would very much appreciate receiving back any new routine, and in particular a portable form for Unix. I have based my Sun routine on a Fortran-callable system-provided routine DTIME, which appears not to be standard (at least not on HP 9000, but works unchanged on DEC Ultrix under the DEC Fortran compiler), and on the other hand I find extremely painful to write a Fortran-callable C routine, due to the unreasonable awkwardness of argument passage and the annoying linking conventions.

The time measuring routine shall be called CPUCLK and has only one CHARACTER*(*) argument. When this argument is equal to the character string 'START' (uppercase) the routine shall initialize (that is store the current time(s) and return). For whatever other value of the argument, the routine shall read the current time(s), compute the difference with the previous START call, and output to the standard output (in whatever format you like) the argument string and the time values.

The routines provided (identical in each of the source files) can be used as examples.

step b: insert the time measuring routine in the source files

For **each** of the three Fortran files, either :

go at the end, and append there your newly written routine

or

uncomment the routine appropriate for your system (just remove all C's in column one ... but be careful not the uncomment the comments, for safety they have a *double* C). All such routines are grouped together at the end, clearly marked.

Make sure all the other versions of CPUCLK are duly commented out.

step c: set up system dependencies

There are system dependencies in the way of opening files, and possibly memory size limitations. I hope all of your Fortran are able to read and write from the standard output (otherwise you'll have to replace all READ(*,*) and WRITE(*,*) with the appropriate logical unit reference). For the rest the code shall be portable (to run on mini's and workstations it has required to me only two marginal changes from the original IBM code, one concerning the largest argument for the exponential, and the other one the usage of *named* BLOCK DATA, while a few problems more occured on PC's, and are described below). Here is a list of typical actions :

in BENCHMAT

locate the **two** groups of OPEN statements and uncomment the one relevant for your system (or add your own). They are marked with extended comments like CIBM, CVAX etc. Make sure the others are commented. This has to be done in two places in the file.

There are also two other places where I had to change the original code in order to run on PC's (for some reason a character comparison of the form A.EQ.'*' fails, and has to be replaced with one of the form A.EQ.CHAR(42)). The latter form should run on any machine which supports ASCII (hence not IBM mainframes), however I used the former form on all machines but PC's. There is a couple of statements preceded by a comment line marked CDOS (and this occurs in two places) : decomment the form appropriate for your case and comment the other out).

<u>in BENCHFIT</u>

similar to above, there are **two** places where the appropriate OPEN statement has to be uncommented and the other commented out.

In order to run on PC's I also had to comment out a character argument in a COMMON block (this argument was unused anyhow), as IBM Professional Fortran is very strict about mixing character and non-character in COMMON blocks. I also had to replace a little piece of code (a GOTO check taken straight from Bevington's book in CURFIT) with an IF-THEN-ELSE, as the original code caused a compiler inconsistency with IBM PC Professional Fortran.

Note that however there are still unresolved linkage problems (memory or stack, or too many levels of subroutines) in this program when linked on a PC, and it is impossible to have the fitting loop run succesfully (system hangs up).

<u>in BENCHDFT</u>

Possibly the program is ready, however if you have memory limitations (or conversely if you do not have any, and want to test it with *very big* arrays) you may want to change the values of the PARAMETER constants MAXDAT and MAXFRE (max number of data points and frequency steps respectively).

I would expect you do not have memory problems with the matrix (416*49) in BENCHMAT and BENCHFIT. I have run the test also on the HP 1000, and some relevant code is included (commented as CHP).

step d: compile and link

As all routines required by each of the programs are in the same source file, compilation and linkage should be straightforward.

You may want to repeat compilation under different optimization level to test how these affect the execution times.

step e: preliminary go

Before running BENCHFIT you shall run BENCHMAT **once**. If you want you may run BENCHMAT more times under different conditions (e.g. writing over an existing binary file, or deleting it and recreating it anew) to test the speed.

step f : go

You should run the tests on an empty system, if possible, otherwise you may have to run them a few times more to get an average (or take the quickest execution, according to your system load).

Run BENCHDFT a few times, try with different values for the data points and frequency steps (it should scale linearly with both), until you are satisfied with the results

Run BENCHFIT once or twice. May be try on different terminals, or redirecting the output on disk. The output is (for each iteration) a listing of the values of the parameters (chi-squares, hydrogen column density in units of 10^{21} cm^{-2}, normalization and temperature for Bremsstrahlung and blackbody). Check the final results (after 14 iterations) are : χ^2 of 119.3 for 43 dof, 2.083, 4.818, 4.466, 0.2383 1.195 (in my experience they are remarkably reproducible from machine to machine).

If you want, go back to step d and recompile with a different optimization level (it is not necessary to rerun BENCHMAT).

4. The tests performed

4.1 The machines

I have run the benchmarks on the following machines (in all cases the system was empty or minimally loaded, with the exception of the IBM). A few notes are given below.

```
a) IBM 3090 (without vector facility) with VM/SP
b) VAX 8250 (RA82 disk) with VMS 5.3
c) Sun 4/110 (SCSI disk) with SunOs 4.0 (unix)
d) HP 1000 F (poor old guy) with RTE 6/VM
e) Olivetti M380 with MS-DOS (wants math coprocessor)
f) Olivetti M300 with MS-DOS (wants math coprocessor)
g) Olivetti M290 with MS-DOS (wants math coprocessor)
h) IBM PC AT     with MS-DOS (wants math coprocessor)
i) Compaq SLT286 with MS-DOS (wants math coprocessor)\
j) Sun 4/330 (SCSI disk) with SunOs 4.1.1 (unix)
k) Sun IPC (over NFS)    with SunOs 4.1.1 (unix)
l)  DECStation  5000/200  (SCSI  disk)  with  Ultrix  4.2
(unix)
m) Sun IPX (SCSI disk) with SunOS 4.1.3, Fortran 1.4
```

I haven't run it on the HP 9000 since Fortran documentation was unavailable and I was unable to set up a Fortran-callable C-routine for time measurement in a way general for Unix systems.

a) on the IBM I have used the VS Fortran compiler with no optimization, and at optimization levels 1,2,3. The system was in condition of load (as indicated by CP IND) with CPU between 14% and 43%, and EXPAN between 8 and 25. Not necessarily the highest CPU corresponds to the highest EXPAN (I have cases of EXPAN 20 CPU 42%, EXPAN 25 CPU 16% and EXPAN 8 CPU 22%). EXPAN once used to express the "time expansion factor" (elapsed vs CPU time), but currently it looks like that the time as measured is virtually unaffected by the external conditions.

The programs require each its own EXEC file.

I have measured elapsed, virtual CPU and total CPU times, using a locally provided TTIMER routine.

b) on the VAX I have used the VAX Fortran compiler with and without optimization. I have run the tests directly or via a self-set host (to make a written log). The latter (as well as the use of local printers) has no effect on the time measured (this applies to BENCHFIT which is the only program doing some amount of terminal output).

I have measured elapsed and CPU times as provided by LIB$SHOW_TIMER (using LIB$INIT_TIMER for initialization).

c-j-k) on the Sun I have used the Sun f77 compiler with no optimization and with levels -O1,-O2,-O3. In the case of BENCHFIT I have tried running it in my normal Sunview desktop (quite crowded of many windows, including clocks and performance meters), in a Sunview desktop with a single window, and on the bare console, as well as redirecting output to file. With some surprise, all times are comparable, *but the one using the console*, which is unexpectedly slower (obviously the console driver has a larger overhead).

I have measured "total", "user" and "system" times as supplied by the Fortran-callable routine dtime.

The test was originally done on a Sun 4/110 (c). It has been repeated after the upgrade of the same machine to 4/330 (j) with the same version of the compiler (1.2). Both machines are SPARC *sun4* kernel architecture.

The SUN IPC (k) is instead a SPARC *sun4c* kernel architecture. Due to unavailability of the Fortran compiler on this machine, the executable generated on the 4/330 was used. The executable and the data were residing on the 4/330 on a disk NFS-mounted to the IPC.

The test was repeated using the new compiler (1.4) on a Sun IPX (m), still with a SPARC *sun4c* kernel architecture, under operating system revision 4.1.3.

d) on the HP 1000 the test requires usage of EMA and some simple changes to the code. An SL,6,luterm is also necessary to direct the standard output to the terminal.

The time measured (via call EXEC(11)) is elapsed time.

e-i) on the IBM-compatible PCs with MS-DOS I have used the IBM Professional Fortran (PROFORT) compiler (which requires the presen ce of a mathematical coprocessor) and the IBM PC limker. There are two warnings in this respect. Concerning BENCHMAT and BENCHFIT, the run time option /R 1664 shall be used to force the maximum record length to a value needed for the matrix binary i/o. Secondly, the main fitting loop of BENCHFIT does not work (the system hangs up and an hard reset is necessary) : this is possibly due to the large number of nested routine calls (stack problems ? the lowest routine is called, but never returns) or to the complex common block orga nization. As a solution of this problem implies rearrangement of the code, and this makes the test no longer comparable with the other machines, I have dispensed running this part of the benchmark.

The times measured are elapsed time, obtained via the PROFORT GETTIM routine.

j-k) These tests, done after one year from the original issue, are described above together with c.

l) on the DECstation I have used the DEC Fortran f77 compiler (1.3) with no optimization and with levels -O1, -O2, -O3. The code from the Sun was used unchanged. All tests ran in a DECwindows Decterm window.

m) these tests (done in Dec 1992) were done using the Sun Fortran 1.4 compiler, in order to test the performance of the Sun IPX, as well as the efficiency of the new compiler on the 4/330 and IPC platforms. The programs were compiled as -Bstatic, to prevent using Fortran shared libraries (these reside on an NFS disk and may slow down the DFT test by a factor 2, while the other tests are nearly unaffected). Formatted i/o looks twice as fast with the new compiler.

4.2 Results of the tests

The results are summarized in the tables in the next pages.New results are indicated by a single or double bar on the right margin.

For IBM the "total CPU time" is indicated
For VAX the CPU time is indicated
For Suns and DECstation the "total" (user+system) time is indicated
For HP the elapsed time (in condition of no load) is indicated
For all PCs elapsed time is indicated.

BENCHDFT

The results given are all for 100 frequency steps, and a limited range of number of data points. All times appear to scale linearly with both quantities (this might not be apparent for the shorter times, or the fastest machines, which are obviously dominated by overheads in this regime). Times given are data generation (DG) and Fourier transform (TF), the number is the number of points.

System	DG 100	1000	10000	FT 100	1000	10000
IBM no opt	0.006	0.009	0.047	0.071	0.688	6.948
IBM opt 1	0.006	0.011	0.044	0.065	0.636	6.339
IBM opt 2	0.006	0.009	0.040	0.059	0.581	5.851
IBM opt 3	0.007	0.010	0.040	0.058	0.573	5.745
VAX no opt	0.03	0.13	1.27	1.65	16.73	170.45
VAX opt	0.02	0.14	1.14	1.47	16.27	152.36
Sun 110 no opt	0.00	0.05	0.58	0.84	8.28	83.94
Sun 110 -O1	0.01	0.06	0.59	0.90	8.48	85.58
Sun 110 -O2	0.00	0.05	0.58	0.81	8.07	81.87
Sun 110 -O3	0.01	0.06	0.58	0.83	8.10	82.60
Sun 330 no opt	0.00	0.01	0.15	0.24	2.38	24.49
Sun 330 -O1	0.00	0.02	0.15	0.24	2.37	24.24
Sun 330 -O2	0.00	0.02	0.14	0.23	2.24	23.16
Sun 330 -O3	0.00	0.01	0.14	0.23	2.28	23.47
Sun IPC no opt	0.01	0.02	0.21	0.28	2.76	28.52
Sun IPC -O1	0.01	0.02	0.22	0.30	2.94	30.38
Sun IPC -O2	0.01	0.02	0.20	0.32	3.16	32.53
Sun IPC -O3	0.01	0.02	0.21	0.32	3.19	32.99
Sun IPX no op 1.4	0.00	0.01	0.11	0.16	1.61	16.76
Sun IPX -O1 1.4	0.00	0.01	0.10	0.16	1.61	16.70
Sun IPX -O2 1.4	0.00	0.02	0.09	0.15	1.54	15.58
Sun IPX -O3 1.4	0.01	0.01	0.11	0.15	1.47	15.61
DEC no opt	0.00	0.008	0.074	0.117	1.144	11.706
DEC -O1	0.00	0.012	0.086	0.145	1.359	13.886
DEC -O2	0.00	0.008	0.070	0.113	1.098	11.202
DEC -O3	0.00	0.008	0.074	0.117	1.137	11.624
HP 1000	0.02	0.20	NA	3.67	36.38	NA
Olivetti M380	0.01	0.10	1.10	2.20	21.81	220.75
Olivetti M300	0.06	0.16	1.92	3.73	36.52	369.37
Olivetti M290	0.06	0.43	NA	10.77	106.18	NA
IBM PC AT	0.05	0.98	NA	21.48	212.12	NA
Compaq SLT286	0.06	0.39	NA	8.08	79.70	NA

BENCHFIT (and **BENCHMAT**)

The times given are as follows: ASCII in, the input of the ASCII data used to rebuild the matrix in BENCHMAT (see 2.3 for details); Mat out, the time to output (and eventually create, when the overhead is relevant) the binary matrix file (49 records of 416 REAL*4) in BENCHMAT; Spec in and Mat in, the time spent in BENCHFIT to read in the ASCII spectrum (48 formatted records) and the binary matrix respectively; Fitting, the time spent in BENCHFIT for 14 CURFIT iterations and the relevant terminal output.

Note that it has not been possible to run the complete BENCHFIT program on PCs due to unresolved linkage problems.

System	ASCII in	Mat out	Spec in	Mat in	Fitting	
IBM no opt	0.233	0.076	0.012	0.048	8.539	(1)
IBM opt 1	0.215	0.065	0.013	0.048	6.819	
IBM opt 2	0.203	0.054	0.012	0.042	4.038	
IBM opt 3	0.213	0.056	0.012	0.041	4.004	
VAX no opt	5.19	1.11	0.24	1.46	144.01	(2)
VAX opt	4.93	0.24	0.23	0.35	96.80	
Sun 110 no opt	4.61	0.61	0.08	0.93	53.29	(4)
Sun 110 -O1	4.54	0.58	0.08	0.84	49.10	
Sun 110 -O2	4.50	0.51	0.08	0.70	34.73	
Sun 110 -O3	4.52	0.48	0.07	0.78	34.52	
Sun 330 no opt	2.86	0.26	0.05	0.41	20.10	
Sun 330 -O1	2.63	0.27	0.05	0.40	18.20	
Sun 330 -O2	2.55	0.22	0.06	0.34	12.33	
Sun 330 -O3	2.56	0.22	0.05	0.35	11.31	
Sun IPC no opt	3.11	0.30	0.07	0.43	21.50	
Sun IPC -O1	3.03	0.28	0.06	0.43	19.46	
Sun IPC -O2	3.03	0.24	0.07	0.37	13.00	
Sun IPC -O3	3.03	0.24	0.07	0.38	12.97	
Sun IPX no op 1.4	0.77	0.15	0.01	0.23	15.89	
Sun IPX -O1 1.4	0.76	0.16	0.02	0.27	12.67	
Sun IPX -O2 1.4	0.74	0.14	0.01	0.25	9.14	
Sun IPX -O3 1.4	0.73	0.14	0.01	0.21	8.67	
DEC no opt	0.617	0.23	0.04	0.266	7.043	
DEC -O1	0.598	0.246	0.047	0.254	19.76	
DEC -O2	0.582	0.223	0.039	0.223	8.144	
DEC -O3	0.547	0.219	0.043	0.219	7.109	
HP 1000	30.80	18.77	0.62	10.62	427.35	(3)
Olivetti M380	5.06	4.06	0.11	1.64	NA	
Olivetti M300	10.38	4.28	0.22	2.91	NA	
Olivetti M290	16.43	6.20	0.11	4.78	NA	
IBM PC AT	32.79	10.99	NA	NA	NA	
Compaq SLT286	16.86	6.10	0.38	5.77	NA	

Notes:

(1) the Mat out time indicated for IBM is for the case the binary file has already been created; if formatting of 49 recs is required, the time becomes 0.116 s. Note also the elapsed time for ASCII in and Mat out was of the order 2 and 3-4 s respectively.

(2) the Mat out time indicated for VAX is in the case the binary file has already been created. If creation is required such time looks very similar (1.18 and 0.30 s for the two cases shown).

(3) the Mat out time indicated for HP 1000 is in the case the binary file has already been created. However file creation with the correct number of records (49) is just slightly longer (18.88 s), while if no preallocation is made is 19.95 s (the file is extended 13 times).

(4) in the case of the Sun the fitting loop is affected by terminal output speed. For default (no opt) compilation the times given are for Sunview (quite insensitive to number of windows and overall load), but becomes as high as 66-77 s for console output, and as low as 48 s if output is suppressed.

Relative performance

The relative performance assumes as unity the case of the IBM (not optimized). As it is apparent from the above tables the effect of the various levels of optimization is different on the various machines and in the different contexts (i/o is certainly the less affected, but its effect is also different on pure CPU tasks, like the Fourier transform, and mixed tasks involving a number of routine calls, like the CURFIT loop). The relative performance does not show up clearly at lower regimes, possibly due to the different weight of overheads (compare the data generation loops for the BENCHDFT program, where the slowest ma chines are faster for the small number of data points, while for the larger number of points they scale as for the Fourier transform) : therefore the DG data from BENCHDFT have been omitted (while the FT data shown are averaged).

System	DFT	ASCII in	Mat out	Spec in	Mat in	Fitting
IBM no opt	1	1	1	1	1	1
IBM opt1	0.92	0.92	0.85	1	1	0.80
IBM opt 2	0.84	0.87	0.71	1	0.88	0.47
IBM opt 3	0.83	0.91	0.73	1	0.85	0.47
VAX no opt	24.02	22.27	14.61	20.00	30.42	16.86
VAX opt	22.09	21.15	3.16	19.17	7.29	11.33
Sun 110 no opt	11.98	19.78	8.02	6.66	19.38	6.24
Sun 110 -O1	12.44	19.48	7.63	6.66	17.50	5.75
Sun 110 -O2	11.64	19.31	6.71	6.66	14.58	4.07
Sun 110 -O3	11.78	19.40	6.31	5.83	16.25	4.04
Sun 330 no opt	3.45	12.27	3.42	4.17	8.54	2.35
Sun 330 -O1	3.44	10.13	3.55	4.17	8.33	2.13
Sun 330 -O2	3.28	10.94	2.89	5.00	7.08	1.44
Sun 330 -O3	3.31	10.94	2.89	4.17	7.29	1.32
Sun IPC no opt	4.02	13.35	3.95	5.83	8.96	2.52
Sun IPC -O1	4.29	13.00	3.68	5.00	8.96	2.34
Sun IPC -O2	4.59	13.00	3.16	5.83	7.71	1.52
Sun IPC -O3	4.63	13.00	3.16	5.83	7.92	1.52
Sun IPX no op 1.4	2.33	3.30	1.97	0.84	4.79	1.86
Sun IPX -O1 1.4	2.33	3.26	2.10	0.84	5.63	1.48
Sun IPX -O2 1.4	2.19	3.17	1.84	0.84	5.28	1.07
Sun IPX -O3 1.4	2.16	3.13	1.84	0.84	4.37	1.01
DEC no opt	1.77	2.65	3.03	3.33	5.54	**0.82**
DEC -O1	2.01	2.57	3.24	3.92	5.29	**2.31**
DEC -O2	1.60	2.50	2.93	3.25	4.65	**0.95**
DEC -O3	1.66	2.35	2.88	3.58	4.56	**0.83**
HP 1000	52.28	132.18	246.97	51.66	221.25	50.05
Olivetti M380	31.48	21.71	53.42	9.17	34.16	NA
Olivetti M300	52.92	44.54	56.31	18.33	60.62	NA
OlivettiM290	153.00	70.51	81.57	9.17	99.59	NA
IBM PC AT	305.42	140.72	144.61	NA	NA	NA
Compaq SLT286	114.83	72.36	80.26	31.67	120.21	NA

Addendum

Further tests have been made on a DECstation 5000/240, a Personal DECstation 5000/33 (both with Fortran 3.2), and in the near future will be made on a Sun Sparc 10. The additional results are reported in the tables here below.

BENCHDFT

System	DG 10000	FT 10000
DEC 5240 no opt	0.066	11.20
DEC 5240 -O1	0.051	8.54
DEC 5240 -O2	0.047	7.00
DEC 5240 -O3	0.047	6.96
DEC 5033 no opt	0.078	13.54
DEC 5033 -O1	0.066	10.39
DEC 5033 -O2	0.059	8.43
DEC 5033 -O3	0.055	8.50

BENCHFIT

System	Spec in	Mat in	Fitting
DEC 5240 no opt	0.039	0.227	17.97
DEC 5240 -O1	0.039	0.141	12.28
DEC 5240 -O2	0.041	0.125	5.10
DEC 5240 -O3	0.035	0.125	4.45
DEC 5033 no opt	0.070	0.426	21.80
DEC 5033 -O1	0.074	0.266	15.19
DEC 5033 -O2	0.066	0.242	6.37
DEC 5033 -O3	0.070	0.246	5.50

Relative performance

System	DFT	Spec in	Mat in	Fitting
DEC 5240 no opt	1.61	3.25	4.7	2.1
DEC 5240 -O1	1.22	3.25	2.9	1.4
DEC 5240 -O2	**1**	3.41	2.6	**0.60**
DEC 5240 -O3	**1**	2.91	2.7	**0.52**
DEC 5033 no opt	1.94	5.8	8.9	2.6
DEC 5033 -O1	1.41	6.2	5.5	1.77
DEC 5033 -O2	1.21	5.5	5.0	**0.75**
DEC 5033 -O3	1.22	5.8	5.1	**0.64**