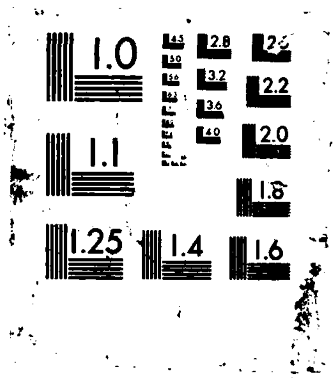
AD-A176 838 FAST FOURIER TRANFORMATION ALGORITHMS: EXPERIMENTS WITH MICROCOMPUTERS(U) SACLANT ASW RESEARCH CENTRE LA SPEZIA (ITALY) B W CONOLLY ET AL. JUL 86 SACLANTCEN-SM-182 1/1

UNCLASSIFIED F/G 12/1 NL


1.0
2.8
3.2
3.6
4.0
2.2
2.0
1.1
1.8
1.25
1.4
1.6

SACLANTCEN Memorandum SM - 182

SACLANTCEN Memorandum SM - 182

AD-A176 838

SACLANTCEN Memorandum SM - 182
(Final Version)

SACLANT ASW RESEARCH CENTRE MEMORANDUM

SACLANT ASW RESEARCH CENTRE

FAST FOURIER TRANSFORMATION ALGORITHMS: EXPERIMENTS WITH MICROCOMPUTERS

by
Brian W. CONOLLY
Ole F. HASTRUP

JULY 1986

Supersedes AD-A161915

This document has been approved for public release and sale; its distribution is unlimited.

NORTH ATLANTIC TREATY ORGANIZATION

SACLANTCEN
LA SPEZIA, ITALY

This document is released to a NATO Government at the direction of the SACLANTCEN subject to the following conditions:

1. The recipient NATO Government agrees to use its best endeavours to ensure that the information herein disclosed, whether or not it bears a security classification, is not dealt with in any manner (a) contrary to the intent of the provisions of the Charter of the Centre, or (b) prejudicial to the rights of the owner thereof to obtain patent, copyright, or other like statutory protection therefor.

2. If the technical information was originally released to the Centre by a NATO Government subject to restrictions clearly marked on this document the recipient NATO Government agrees to use its best endeavours to abide by the terms of the restrictions so imposed by the releasing Government.

Published by

SACLANTCEN MEMORANDUM SM-182
(Final Version)

NORTH ATLANTIC TREATY ORGANIZATION

SACLANT ASW Research Centre
Viale San Bartolomeo 400,
I-19026 San Bartolomeo (SP), Italy.

tel: national 0187 540111
international + 39 187 540111

telex: 271148 SACENT I

FAST FOURIER TRANSFORMATION ALGORITHMS
EXPERIMENTS WITH MICROCOMPUTERS

by

Brian W. Conolly and Ole F. Hastrup

July 1986

This memorandum has been prepared within SACLANTCEN Underwater Research Division as part of Project 18.

O.F. Hastrup

O.F. Hastrup
Division Chief

CONTENTS

INTRODUCTION

~~We~~ describes experiments intended to exploit the potential of modern microcomputers for harmonic analysis. ~~Our~~ findings are a contribution to the discussion of how far modern microcomputers can complement, compete with, and, in certain circumstances, substitute for the mainframe.

Harmonic analysis is fundamental to signal processing which, in turn, has many applications both in civilian and military contexts. The publication of so-called Fast Fourier Transform (FFT) algorithms revolutionised digital analysis: results which had previously required many hours of computation could be obtained in minutes. Microcomputers can not yet compete with mainframes in terms of speed but they do have the important advantages of portability and lower cost. ~~Our~~ experience shows that equipment exists which combines the advantages of speed and an accuracy adequate for contaminated data, with portability, availability, and versatility that are characteristic of the microcomputer.

~~Our~~ attention is confined to the Fast Fourier Transform about which so much has been written. We define for data length t, the discrete Fourier transform, by

$$A_s = \frac{1}{t}\sum_{r=0}^{t-1} a_r \exp(-2\pi rs/t), \quad (s=0,1,2,\ldots,t-1). \qquad \text{(Eq. 1)}$$

The "time domain" function, a_r, can be recovered from its "frequency domain" counterpart, A_s, by the formula

$$a_r = \sum_{s=0}^{t-1} A_s \exp(2\pi rs/t), \quad (r=0,1,2,\ldots,t-1). \qquad \text{(Eq. 2)}$$

Although t is equal to a power of 2 in the experiments the conclusions are almost entirely independent of this condition.

1 TEST FUNCTIONS

For experimental purposes it is convenient to use test functions, that is, functions a_r with a known, discrete Fourier transform A_s. Such functions are given in [1]. The functions, TF1, TF2, and TF3, were used and are described below.

	a_r	A_s
(TF1)	$\exp(2\pi r^2 i/t)$	$(1+i)(1+i^{-2s-t})\exp(-2\pi s^2 i/4t)/(2(t)^{1/2}$

This pair generalises the so-called Gauss Sum, known in Number Theory and signal processing, and applies to the "chirp" transform. Both a_r and A_s are complex.

	a_r	A_s
(TF2)	r/t	$(1-1/t)/2 \quad (s=0)$ $(-1+i\cot(\pi s/t))/(2t) \quad (0<s<t-1)$.

This function is real in the time domain and its transform is complex.

	a_r	A_s
(TF3)	$r(1-r/t)/t$	$(1-1/t^2)/6 \quad (s=0)$ $-1/(2t^2\sin^2\pi s/t) \quad (0<s<t-1)$.

This function and its transform are both real. In some cases a sinusoid was used as input.

Test functions have certain benefits. The input function, a_r, can be calculated to the full accuracy given by the computer; A_s can be obtained numerically by use of the algorithm. Then A_s can be compared in various ways with the values calculated to full machine accuracy from the inversion formula. This supplements the commonly used procedure of getting A_s numerically by the algorithm, inverting numerically, and comparing the result with a_r.

The choice of test functions was made to permit experimentation with versions of the algorithms that were designed for full complex input and output as well as modifications of the algorithms for real time domain data and real and complex results in the frequency domain. The modifications usually offer substantial savings in time.

2 COMPUTERS

The two types of personal computers that were used in the experiments were suitable for scientific work. They included the Apple (Series II) with a Synertek 6502 processor and an 8-bit word and a portable IBM PC, with an Intel 8088 microprocessor and 8087 coprocessor with a 16-bit word; this latter system is a powerful combination. A Microsoft BASIC compiler and the standard Apple PASCAL system were available for the Apple. The IBM PC used the Borland 2.0 Turbo-87 PASCAL compiler and the Microsoft 3.20 FORTRAN compiler; both use the 8087 coprocessor. The Apple language system also offers FORTRAN under the PASCAL operating system but was not used. Nevertheless, comparisons made with PASCAL programs give a sufficient indication of the relative speed of the two PC systems.

3 ALGORITHMS

The FFT algorithms used in the experiments are all based on Cooley-Tukey [2], or Sande-Tukey [3], sometimes with modifications (see Singleton [4]) for recursive calculations of sines and cosines. Because t is a power of 2 the basis of the algorithms is the expression of r and s in the scales of 2 or 4, combined if necessary. Because of a substantial variation in speed between the algorithms, most of the experiments were limited to the fastest performers. Table 1 lists the algorithms under the names we have used, the original language, and the reference.

TABLE 1

ALGORITHMS AND THEIR SOURCE

Name	Original Language	Reference
MCGW/Singleton	ALGOL	[5], [11]
FFT/10	BASIC	[6]
BRENNER modified	FORTRAN	[7]
MONRO 4/5	FORTRAN	[8]
MONRO modified	FORTRAN	[9]
FOURT	FORTRAN	[10]

The MONRO algorithms were translated into both BASIC and PASCAL. FOURT was translated into BASIC.

4 PRELIMINARY TIMINGS

Table 2 gives timings to the nearest second for the Apple II. Test Function 1 is used. The programs are MCGW, FFT/10, MONR04, MONR05 and FOURT. MONR04 is Algorithm AS83 taken from [8]. MONR05 is a version of MONR04 that uses do-loops for unscrambling as suggested in [3]. D.M. MONRO claimed this version is faster, but application of MONR05 to the Apple II presented some difficulties; in the end, the speed gain, if any, was insignificant.

TABLE 2

APPLE II TIMINGS TEST FUNCTION 1

Timings (s)

Length t	MCGW	FFT/10	MONR04		MONR05		FOURT
	PASCAL	Compiled BASIC	Compiled BASIC	PASCAL	Compiled BASIC	PASCAL	Compiled BASIC
32	10	4	1	3	1	3	1
64	25	11	3	5	3	6	3
128	62	24	8	14	8	14	7
256	146	56	17	31	16	30	14
512	341	128	38	69	37	69	32
1024	783	284	80	148	79	144	70

Because speed is our main interest we shall say no more abcut MCGW and FFT/10. Thus the MONRO series and FOURT remain the only serious contenders in terms of speed. Although both the MONRO and FOURT methodologies seem identical, FOURT has the speed advantage because of some factor not yet identified.

Table 3 gives similar timings for runs in the MONRO series, BRENNER modified, and MCGW using the IBM PC, all with TF1 (Eq. 1). The compilers provided options to improve performance, as noted, for which a penalty in compiling time has to be paid. The times relate exclusively to 1024 point transforms.

TABLE 3

IBM PC + 8087 COPROCESSOR TIMINGS

(All timings relate to 1024 point transforms)

Test Function 1

Algorithm	Language	Precision	Compiler Options	Timings (s)
MONRO5	FORTRAN	Single	Yes	3.7
MONRO5	FORTRAN	Double	Yes	4.5
MONRO5	FORTRAN	Single	No	5.9
MONRO5	PASCAL	Double	No	13.6
BRENNERmod	FORTRAN	Single	Yes	14.3
BRENNERmod	FORTRAN	Single	No	16.4
MCGW/Singleton	PASCAL	Double	No	36.5

In comparison the Brenner modified algorithm performs a 1024 point transform on a Univac 1106 with an FPS pipeline processor in 0.006 seconds in double precision FORTRAN.

FOURT was not run; however it would probably be faster than MONRO5 under similar conditions. However the differences in timing would be measured in nothing more significant than tenths of seconds.

Other mainframe timings of the MONRO algorithms are given in [8] and [9].

5 ACCURACY

So far we have not discussed accuracy systematically or in depth, for two reasons. First, the other algorithms had slower speeds when compared with the MONRO series and FOURT and were not further tested because our interest was in speed. Second inspection of the output indicates that the accuracy attained by faster algorithms on the IBM PC was as high as the manufacturer or user would wish when dealing with data contaminated by noise. This conclusion is based on a forward and backward transformation of TF1, comparing a_r with the numerical inverse of its algorithmically obtained transform.

The first three series of the Apple II experiments were carried out with the MONRO 4 programs in Microsoft Compiled BASIC only.

Series 1

In this series the procedures were as follows:

(i) Calculate the input values for TF1 of a_r and the modulus $|a_r|$ (which is theoretically unity in this case).

(ii) Calculate by the algorithm the transform $\hat{A}_s$.

(iii) Calculate by the algorithm the inverse of $\hat{A}_s$, say $\hat{a}_r$, *and form* $|\hat{a}_r|$.

(iv) Calculate

$$m_1 = \frac{1}{t} \sum_{r=0}^{t-1} (|a_r| - |\hat{a}_r|)$$

and

$$m_2 = (s_2 - m_1^2)^{1/2}$$

where

$$s_2 = \frac{1}{t} \sum_{r=0}^{t-1} (|a_r| - |\hat{a}_r|)^2 .$$

m_1 and m_2 can be interpreted, respectively, as the mean and standard deviation of the difference between a_r and $\hat{a}_r$.

Series 2

The same procedures were used for this series of experiments which used the Apple II. However, m_1 and m_2 are based on a comparison of $R1A_s$, the real part of the theoretical inverse, and $R1\hat{A}_s$, the real part of the algorithmically obtained inverse. Thus,

$$m_1 = \frac{1}{t} R1 \sum_s (\hat{A}_s - A_s),$$

and $m_2 = (s_2 - m_1^2)^{1/2}$,

where $s_2 = \frac{1}{t} \sum_s (R1\hat{A}_s - R1A_s)^2$.

Series 3

In this series max $(|A_s - \hat{A}_s|)$ is calculated.

Series 4

FOURT in Compiled BASIC was run using TF1. A criterion involving real and imaginary parts was evaluated. The BASIC realisation of FOURT retained the Fortran-type storage for complex data in an array B of length 2t. Thus $b_{2r-1} = R1a_r$, $b_{2r} = Ima_r$, $r=0,1,2,....(t-1)$. With obvious notation, max $[\,|\hat{b}_k - b_k|\,]$ is calculated for $1 \leq k \leq 2t$.

The results of the Series 4 of Apple II experiments are given in Table 4.

TABLE 4

CALCULATIONS OF ACCURACY

Length t	Series 1		Series 2		Series 3	Series 4
	$m_1 \times 10^{-10}$	$m_2 \times 10^{-10}$	$m_1 \times 10^{-10}$	$m_2 \times 10^{-6}$	max $\times 10^{-9}$	max $\times 10^{-9}$
32	0.73	13.5	2.6	7.4	5.7	5.9
128	2.15	16.6	-0.83	5.9	18.2	11.4
512	1.35	43.3	-2.1	4.3	33.8	36.2
1024	9.9	29.4	0.86	3.6	57.5	59.7

Table 4 shows a very satisfactory performance for both algorithms and microcomputer on the basis of several criteria. The directly comparable Series 3 and Series 4 show little difference. Note that in Series 2, m_2 decreases as t increases.

More discussion in the context of mainframes is available in [8], [9] and many other sources. Our work seems to be the first attempt to review the accuracy of the algorithms in the context of the microcomputer.

6 REAL INPUT DATA

The final part of this report deals only with timings. Table 5 gives values to the nearest second for the Apple II and test functions TF2 (Eq. 2) and TF3 (Eq. 3).

TABLE 5

APPLE II TIMINGS: REAL a_r

Timings (s)

	TF2			TF3			
Program	MONR04	MONR07	FOURT	FFT/10	MONR04	MONR08	FFT/10
32	1	1	2	6	1	2	4
64	3	3	3	10	3	3	11
128	7	6	6	24	8	6	26
256	17	14	14	56	17	13	58
512	37	28	31	127	36	29	128
1024	78	61	66	279	77	60	290

The following notes apply to the Table 5 data:

- All runs were made in Microsoft Compiled BASIC.
- MONR04 is the full, complex algorithm; both MONR07 and 8 are faster versions adapted to real time-domain data. All three algorithms are described in references [8] and [9].
- FOURT has been tested only with TF3.
- The algorithms were not translated into PASCAL because the relative performance of the Apple II in Compiled BASIC vis-a-vis PASCAL can be seen in Table 2.

The following comments apply to the Apple II timings:

- MONR04 performs at about the same speed with real input as when the input is complex. Specially adapted MONR07 and MONR08 offer significant improvements.
- Under TF3, FOURT is slightly slower than MONR07. However the authors of FOURT claim that it can be made to run up to 40 percent faster with real data. This claim should be tested.
- Like MONR04, FFT/10 is much the same as with a complex input.
- On the IBM PC, the special MONRO algorithm for real data in its original FORTRAN and for a t=1024 transform requires 2.8 seconds in single precision and 3.6 seconds in double; in each case special compiler optimisation options are used.

CONCLUSIONS

Although unsystematic, the results of the experiments support the view that the modern personal computer has an increasingly important role to play in signal processing applications in the field and in the laboratory. When a sophisticated requirement calls for thousands of Fourier transforms in seconds (or even minutes) the mainframe continues to hold its own. The attractive attributes of the personal computer are its availability, price, size, and portability as compared with older systems.

The Apple II and the IBM PC microcomputers performed well in implementing efficient algorithms on the demanding complex Fast Fourier Transform test function (TF1). Both personal computers can be applied to a wide range of practical situations including the use of the FFT as an approximation to the Fourier integral.

For 8 and 16 bit computers we have found the Apple II and IBM PC to be excellent tools in a wider field of scientific applications that include traditional numerical analysis and statistical data processing and inference, and Monte Carlo simulation. They provide scientists with a fast, reliable and accurate tool at a reasonable price. This comment applies also to other professional personal computers in the field, although only the Apple II and IBM PC were tested.

Detailed statistics on the speed of various processors ranging from the fastest mainframes to microcomputers are found in [13]. The tests described in that report were conducted in the specialised environment of the solution of dense systems of linear equations; however the absolute comparisons are likely to be valid in a wider context.

REFERENCES

1. CONOLLY, B.W. and GOOD, I.J. (1977) A table of discrete Fourier transform pairs. SIAM Journal of Applied Mathematics, 32,1977:810-822.

2. COOLEY, J.W. and TUKEY, J.W. An algorithm for the machine calculation of complex Fourier series. Mathematics of Computation, 19, 1965:297-301.

3. GENTLEMAN, W.M. and SANDE, G. (1966) Fast Fourier transforms for fun and profit. *In: American Federation* of Information Processing Societies. Conference proceedings, fall joint computer conference, volume 29. Washington D.C., spartan Books, 1966: pp 563-578.

4. SINGLETON, R.C. On computing the fast Fourier transform. Communications of the ACM, 10, 1967: 647-654.

5. MCGREGOR, J.J. and WATT, A.H. PASCAL for Science and Engineering. London, Pitman, 1983. [ISBN 0-273-01889-2]

6. STANLEY, W.D. and PETERSON S.J. Fast Fourier transforms on your home computer. Byte, 3 (12), 1978: 14-25.

7. CLAERBOUT, J.F. Fundamentals of Geophysical data processing with applications to petroleum prospecting. New York, NY, McGraw Hill, 1976. [ISBN 07 011117-0]

8. MONRO, D.M. Complex discrete Fast Fourier transform. Applied Statistics 24, 1975: 153-160.

9. MONRO, D.M. Real discrete Fast Fourier transform. Applied Statistics 25, 1972: 166-172.

10. BRENNER, N.M. Three FORTRAN programs that perform the Cooley-Tukey Fourier transform. Tech. Note 167-2, MIT. Lincoln Lab., Lexington, Mass. 1967. (*Note: We have only* a FORTRAN listing of a very general program attributed to this author, but it is stated to be based also on the work of Charles Rader and Ralph Alter. We have not been able to trace a published *reference for certain*. The program should not be confused with BRENNER modified which is described in [7]).

11. SINGLETON, R.C. (1969) An algorithm for computing the mixed radix fast Fourier transform. IEEE Transactions Audio and Electroacoustics, 17, 1969: 93-103.

12. INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. ACOUSTICS, SPEECH, AND SIGNAL PROCESSING SOCIETY. Digital Signal Processing Committee. Programs for digital signal processing. Piscataway, N.J. IEEE Press. 1979. [ISBN 0-87942-127-4].

13. DONGARRA, J. J. (1984) Performance of various computers using standard linear equation software in a Fortran environment. Computer Architecture News, 13, 1985: 3-11.

INITIAL DISTRIBUTION

MINISTRIES OF DEFENCE	Copies
JSPHQ Belgium	2
DND Canada	10
CHOD Denmark	8
MOD France	8
MOD Germany	15
MOD Greece	11
MOD Italy	10
MOD Netherlands	12
CHOD Norway	10
MOD Portugal	2
MOD Spain	2
MOD Turkey	5
MOD U.K.	20
SECDEF U.S.	68

NATO AUTHORITIES	Copies
Defence Planning Committee	3
NAMILCOM	2
SACLANT	10
SACLANTREPEUR	1
CINCWESTLANT/COMOCEANLANT	1
COMSTRIKFLTANT	1
COMIBERLANT	1
CINCEASTLANT	1
COMSUBACLANT	1
COMMAIREASTLANT	1
SACEUR	2
CINCNORTH	1
CINCSOUTH	1
COMNAVSOUTH	1
COMSTRIKFORSOUTH	1
COMEDCENT	1
COMMARAIRMED	1
CINCHAN	3

SCNR FOR SACLANTCEN	Copies
SCNR Belgium	1
SCNR Canada	1
SCNR Denmark	1
SCNR Germany	1
SCNR Greece	1
SCNR Italy	1
SCNR Netherlands	1
SCNR Norway	1
SCNR Portugal	1
SCNR Turkey	1
SCNR U.K.	1
SCNR U.S.	2
SECGEN Rep. SCNR	1
NAMILCOM Rep. SCNR	1

NATIONAL LIAISON OFFICERS	Copies
NLO Canada	1
NLO Denmark	1
NLO Germany	1
NLO Italy	1
NLO U.K.	1
NLO U.S.	1

NLR TO SACLANT	Copies
NLR Belgium	1
NLR Canada	1
NLR Denmark	1
NLR Germany	1
NLR Greece	1
NLR Italy	1
NLR Netherlands	1
NLR Norway	1
NLR Portugal	1
NLR Turkey	1
NLR UK	1

Total initial distribution	248
SACLANTCEN Library	10
Stock	22
Total number of copies	280

END

DATE
FILMED

-87