A new superfast bit reversal algorithm

Article *in* International Journal of Adaptive Control and Signal Processing · December 2002

DOI: 10.1002/acs.718

3 authors, including:

Manuel Rubio Sánchez
King Juan Carlos University
37 PUBLICATIONS **362** CITATIONS

SEE PROFILE

Pedro Gomez
NeuMinNet
348 PUBLICATIONS **3,311** CITATIONS

SEE PROFILE

ACS : 718
pp. 1--5 (col. fig.: NIL)
PROD. TYPE: TYP
3B2
ED: GRACY
PAGN: TERRY SCAN: NIL

INTERNATIONAL JOURNAL OF ADAPTIVE CONTROL AND SIGNAL PROCESSING
Int. J. Adapt. Control Signal Process. 2002; **16**:000–000 (DOI: 10.1002/acs.718)

A new superfast bit reversal algorithm

M. Rubio[1], P. Gómez[1] and K. Drouiche[2,*,†]

[1] *Universidad Politécnica de Madrid, Facultad de Informática Campus de Montegancedo, s/n, 28660 Boadilla del Monte, Madrid, Spain*

[2] *Université de Cergy Pontoise, LPTM, CNRS ESA 8089, Neuville III, 95301 Neuville Sur Oise, France*

SUMMARY

In this paper we present a new bit reversal algorithm which outperforms the existing ones. The bit reversal technique is involved in the fast Fourier transform technique (FFT), which is widely used in computer-based numerical techniques for solving numerous problems. The new approach for computing the bit reversal is based upon a pseudo-semi-group homomorphism property. The surprise is that this property is almost trivial to prove but at the same time it also leads to a very efficient algorithm which we believe to be the best with only $\mathcal{O}(N)$ operations and optimal constant, i.e. unity. Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS: benchmark; bit reverse; fast algorithm; FFT; shuffling data

1. INTRODUCTION

Computing the *Fourier Transform* of a data set is not only easy but very fast to perform using the popular fast Fourier transform algorithm (FFT) involving the *Danielson–Lanczos lemma* and which works properly when the number of available data is a power of 2. Assume that we have $n \in \mathbb{N}^*$ observations of f, a 2π-*periodic*, $\mathbb{L}^2_{[0,\,2\pi]}$ function, say at points $(x_l)_{0 \leqslant l < n}$, uniformly distributed, i.e. $x_l = 2\pi l/n$. Then its Fourier coefficient C_k, at lag $0 \leqslant k < n$, approximated by $C_{k,\,n}$, is obtained discretizing the integral

$$C_k = \frac{1}{2\pi} \int_0^{2\pi} f(x) \mathrm{e}^{-\mathrm{i}kx}\, \mathrm{d}x$$

by the *Riemann* sum

$$C_{k,\,n} = \frac{1}{n} \sum_{l=0}^{n-1} f(x_l) \mathrm{e}^{-\mathrm{i}kx_l}$$

*Correspondence to: Dr K. Drouiche, Universidad Politechnica de Madrid, Facultad de Informática Campus de Montegancedo, s/n 28660 Boadilla del Monte, Madrid, Spain
† E-mail: kd@ptm.u-cergy.fr

Received 19 April 1999
Revised 13 October 2000
Accepted 15 May 2001

but as the sampling is uniform, we have

$$C_{k,n} = \frac{1}{n}\sum_{l=0}^{n-1} f_l e^{-2i\pi kl/n} \tag{1}$$

where we set $f_l = f(x_l)$. Computing the sequence $(C_{k,n})_{0\leqslant k<n}$ requires a number of multiplications of order n^2 which could quickly be seriously disabling if n is large enough. The FFT (see, for more details, References [1–9]), is used to decrease significantly the above quantity to about $n\log_2 n$, where $\log_2 n = \log n/\log 2$. However, to use the most widely used FFTs (see, for more details, Reference [10]), there is a price to pay for that, the sample size, n, has to be a power of 2, i.e. there exists $p \in \mathbb{N}^*$ such that $n = 2^p$ to allow the FFT algorithm to work properly.

Henceforth, we shall assume that $n = 2^p$ for some $p \in \mathbb{N}^*$, now to compute the sequence $(C_{k,n})_{0\leqslant k<n}$, the FFT algorithm needs a preliminary treatment of the data called the 'shuffle' stage which is known as the *bit reversal* procedure; roughly speaking, a special permutation of the indexed data is made, i.e. we first have to scramble the data using the one-to-one mapping [11, 8]:

$$\sigma_p : \mathbb{W}_p \to \mathbb{W}_p$$
$$l \mapsto j = \sigma_p(l)$$

where $\mathbb{W}_p \subset \mathbb{N}$ and $\mathbb{W}_p = \{l \in \mathbb{N}/0 \leqslant l < 2^p\}$ and where $\sigma_p(l)$ denotes the *bit reverse* of j, i.e. if $l \in \mathbb{W}_p$ with $l = \sum_{n=0}^{p-1} \alpha_n 2^n, \alpha_n \in \{0,1\}$, then $\sigma_p(l) = j = \sum_{n=0}^{p-1} \alpha_n 2^{p-n-1}$. It is easy to see that σ_p is involutive and bijective, moreover for all $p \in \mathbb{N}^*$, we have $\sigma_p(0) = 0$ and $\sigma_p(1) = 2^{p-1}$. For example, if $p = 3$ we have the following mapping $\sigma_3(.)$:

Index, i	Binary	Bit reversion	Bit reversed index, $j = \sigma_3(i)$
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

$\sigma_3(\cdot)$

Several numerical techniques exist to compute the indexes $(\sigma_p(l))_{0\leqslant l<2p}$, see References [1–7,10,12]. The most efficient algorithm for computing the bit reversion, which is the fastest and requires the smallest size for storage, is due to Raimund Meyer (1988), which algorithm is available in Van Loan [13]. However, this algorithm does not provide the best efficiency we could expect because it involves testing instructions which delay the computing of $(\sigma_p(l))_{0\leqslant l\leqslant 2p}$, see References [1, 4, 9, 10, 12]. Hereafter, we propose a new algorithm which is test free and moreover outperforms the known ones in terms of the computing speed of the sequence $(\sigma_p(l))_{0\leqslant l<2p}$ moreover it privileges additions, which costs less in terms of computing time, rather than multiplications. For this we provide the following results.

However, before going further, the reader is recommended to appreciate the difficulty of the problem by trying to solve it at this point.

2. MAIN RESULT AND THE ALGORITHM

Let $n_k = 2^k - 1$, $1 < k < p - 1$, note that n_k belong to $\mathbb{W}_p$. We start giving the following definition:

Definition 1

We shall say that $(n_k)_{1<k<p}$ is a node of $\mathbb{W}_p$, if and only if $n_k = 2^k - 1$, $1 < k < p - 1$.

Observe that $\mathbb{W}_p$ has $p - 2$ nodes. The following proposition will provide us the key tool to derive our new algorithm.

Proposition 1

Given $p \in \mathbb{N}^*$, fix n_k, $1 < k < p$, a node of $\mathbb{W}_p$ and let $l \in \mathbb{W}_p$. For all $0 \leqslant l < n_{k-1}$, we have

$$\sigma_p(n_k - l) = \sigma_p(n_k) - \sigma_p(l) \tag{2}$$

where $\sigma_p(n_k) \in \mathbb{W}_p$ is a constant which depends only on k, i.e.

$$\sigma_p(n_k) = \sum_{j=0}^{k-1} 2^{p-j-1} \tag{3}$$

The above proposition claims that the mapping σ_p has a pseudo-homomorphism group property around the nodes n_k, we say pseudo because there is no homomorphism onto $\mathbb{N}$. Observe that this property is only valid around the nodes n_k. This property is the key tool for our proposed algorithm.

Now we finish with our main result which is the algorithm for computing $(\sigma_p(l))_{0 \leqslant l < 2^p}$.

- Initialization: Set first node $n_2 = 2^2 - 1 = 3$, we set $\sigma_p(0) = 0$, $\sigma_p(1) = 2^{p-1}$, $\sigma_p(2) = 2^{p-2}$ and $\sigma_p(3) = \sigma_p(1) + \sigma_p(2) = 3 \times 2^{p-2}$.
- *Do*: for $k = 3$ to $p - 2$, set $n_k = 2^k - 1$ and $\sigma_p(n_k) = \sigma_p(n_{k-1}) + 2^{p-k}$
  - Do for $0 < l < n_{k-1}$, $\sigma_p(n_k - l) = \sigma_p(n_k) - \sigma_p(l)$
- **end**

Exact theoretical number of operations (all types of operations) is given by

$$\underbrace{4}_{\text{Initialization}} + \underbrace{2(p-4)}_{\text{first loop}} + \underbrace{\sum_{k=2}^{p-2}(2^k - 2)}_{\text{second loop}} + \underbrace{2^{p-1}}_{\text{odd terms}} = 4 + 2(p-4) + 2^2(2^{p-3} - 1) - 2(p-4) + 2^{p-1}$$

$$= 2^p = N.$$

3. BENCHMARK

To assess the performance of our new algorithm, we have compared it to two algorithms: the Meyer's algorithm, (see, Reference [13], for more details), and the Evan's algorithm (see, Reference [7], C language source code included). We have performed numerical simulations comparing these algorithms and obtained the following benchmark reported in the table below. It gives the number of operations to compute the bit reversal procedure versus p, where $n = 2^p$. Simulations were performed on a Pentium PC-200 MHz using *C* code.

p	1	2	3	4	5	6	7	8	9	10
ρ_M	2	11	28	70	150	324	662	1364	2746	5560
ρ_E	2	15	36	90	192	415	847	1745	3512	7111
ρ	1	2	4	19	42	81	152	287	550	1069
p	11	12	13	14	15	16	17	18	19	20
ρ_{M}	11 142	22 404	44 834	89 888	179 806	360 028	720 090	144 0984	2882 006	5765 588
ρ_{E}	14 250	28 653	57 339	114 58	229 954	460 440	920 924	1842 875	3685 798	7373 611
ρ	2100	4155	8258	16 457	32 848	65 623	131 166	262 245	524 396	1048 691

Number of operations versus p or $n = 2^p$.

ρ_{M}, ρ_{E} and ρ, respectively, are the number of operations performed to compute the bit reversion using Meyer's, Evan's and our algorithm. Raimund Meyer algorithm is currently used in the famous `radix-2 fft` procedure (including matlab), the fastest one to our knowledge, it is about $\mathcal{O}(n)$ operations, see Van Loan [13]. We have also fit the above data to a linear curve using the least-squares method, the results show that all algorithms are $\mathcal{O}(n)$, ($\rho = an$) operations, but our algorithm has the smallest optimal slope, i.e. unity:

Algorithm	a
Meyer's	5.4958
Evan's	7.0285
New	1.0003

It turns out that our algorithm outperforms Raimund's and Evan's and is more than five times faster.

Remark 1

Take note that initialization is needed only for Meyer's algorithm and Evan's algorithm and not for the new algorithm (however, we do not consider the initialization step for counting the number of operations). Also note that our algorithm is not optimized as several computations of the same index are involved.

4. PROOF OF THE RESULTS

Proof of the Proposition

Although the proof is now trivial, because we pointed out the good property which solves the problem, we write it down here below. Observe that for any $2 \leqslant k < p$, we have

$$n_k = \sum_{j=0}^{k-1} 2^j$$

now let $l < n_{k-1}$, l could be written as

$$l = \sum_{j=0}^{k-1} \alpha_j 2^j, \quad \alpha_j \in \{0, 1\}$$

thus

$$n_k - l = \sum_{j=0}^{k-1} (1 - \alpha_j) 2^j$$

hence we have

$$\sigma_p(n_k - l) = \sum_{j=0}^{k-1} (1 - \alpha_j) 2^{p-j-1}$$

finally, summing properly, we get

$$\sigma_p(n_k - l_k) + \sigma_p(l) = \sum_{j=0}^{k-1} 2^{p-j-1}$$
$$= \sigma_p(n_k)$$

which completes the proof of our theorem. □

The corollary follows immediately.

REFERENCES

1. Blahut RE. *Fast Algorithms for Digital Signal Processing*. Addison-Wesley: New York, 1984.
2. Bracewell R. *The Fourier Transform and its Applications*. McGraw-Hill: New York, 1965.
3. Cooley JW, Tukey OW. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation* 1965; **19**:297–301.
4. Drouiche K. A pseudo homomorphism group property for bit reversal algorithm. *RR*-96-4403, UCP.
5. Duhamel P, Vetterli M. Fast Fourier transforms: a tutorial review. *Signal Processing* 1990; **19**:259–299.
6. Gergkand GD. A guided tour of the fast Fourier transform. *IEEE Spectrum* 1969;41–52.
7. Karp A. Bit reversal on uniprocessors. *SIAM Review* 1996; **30**(1).
8. Lonnie C. *Fundamentals of Digital Signal Processing*. Wiley: New York, 1987.
9. Press WH, Flannery BP, Teukolsky SA, Vetterling WT. Fast Fourier transform. In *Numerical Recipes in Fortran: The Art of Scientific Computing* (2nd edn). Cambridge University Press, Cambridge, England, 1992; 490–529, Chapter 12.
10. Lipson JD. *Elements of Algebra and Algebraic Computing*. Addison-Wesley: Reading, MA, 1981.
11. Stoer J, Bulrisch R. *Introduction to Numerical Analysis*. Springer: New York, 1980.
12. Nussbaumer HJ. *Fast Fourier Transform and Convolution Algorithms* (2nd edn). Springer: New York, 1982.
13. Van Loan C. *Compuational Frameworks for The Fast Fourier Transform*. SIAM: Philadelphia, 1992.
14. Walker JS. *Fast Fourier Transform* (2nd edn). CRC Press: Boca Raton, FL, 1996.

