Implementation of the Generalized FFT on Finite Groups

Student authors: Igor Mihajlović[1], Milan Marković[1], Nenad Andrejević[1], and Milan Djokić[1]
Mentors: Radomir Stanković[2] and Dušan Gajić[2]

Abstract - **FFT is one of the most important algorithms in signal processing and computing. The group theoretical approach to Fourier analysis is useful in solving several optimization problems in these areas. For practical application of this approach, the corresponding FFT algorithms on finite groups are required. It is interesting to explore how different group choice at various steps of FFT affects computational efficiency and function spectra. This paper presents a C/C++ implementation of FFT over finite not necessarily Abelian groups allowing to freely choose the underlying group structure at each step of the FFT. We also provide results and conclusions drawn from experiments over a set of benchmark functions viewed as functions on different groups with group representations selected in different fields, including the field of complex numbers and few preselected finite fields. These results could be helpful in tracing directions for further research in this area.**

Keywords – **FFT, C/C++ implementation, finite non-Abelian groups, experimental results.**

I. Introduction

Fourier analysis is the cornerstone of signal processing and system analysis and, hence, has many applications in modern computational problems. Classical Fourier analysis is defined on the unit circle, on the integers and on the real line. A vast number of different fast algorithms have been developed for such theoretical settings [1].

Group theoretical approach to Fourier analysis has been implicit in many of the classical works and its implicit introduction into the field produced many important theoretical conclusions. This approach creates a powerful platform for the unified approach when applying Fourier transform on signals defined on different algebraic structures that reflect the properties of the modelled phenomenon.

Methods based only on finite Abelian groups has provided many successful applications in signal processing, however, recent studies have shown that this approach does not always yield best performance of related algorithms.[2] Hence, non-Abelian groups have been introduced in order to supplement for disadvantages of the former approach.

In this work we investigate how choice of algebraic structures influences performance of the FFT and sparsity of the Fourier spectra.

Student authors:
[1]Igor Mihajlović, Milan Marković, Nenad Andrejević, and Milan Djokić are with the Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mails: igor.mihajlovic1987@gmail.com, milan@elfak.rs, neca.87@gmail.com, Milan.djokic.87@gmail.com .

Mentors:
[2]Radomir Stanković and Dušan Gajić are with the Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mails: radomir.stankovic@gmail.com, dusan.gajic@elfak.ni.ac.rs.

II. Fourier Transform on Finite Groups

The Fourier transform on finite Abelian and non-Abelian groups can be studied in a unique setting with a classical Fourier transform in the frame of abstract harmonic analysis [2].

From the mathematical topology point of view, the real line R is a locally compact Abelian group and the theory of Fourier analysis can been extended to such groups if the exponential functions used in Fourier analysis on R are replaced by the group representations [2].

Def. 1. Finite-dimensional representation of a finite group G over a field P is a homomorphism

$$R : G \rightarrow GL(n, P)$$

where $GL(n,P)$ is the general linear group, i.e., the group of ($n \times n$) invertible matrices (n is a natural number) with respect to matrix multiplication, with entries in a field P.

It can be proven, see for instance [2], that if representations are unitary and irreducible that they form a complete orthogonal system and, hence, the Fourier transform for a function f on a group G of order g can be defined as follows:

$$\mathbf{S}_f(w) = r_w g^{-1} \sum_{u=0}^{g-1} f(u) R_w(u^{-1}), \quad (1)$$

$$f(x) = \sum_{w=0}^{K-1} Tr(\mathbf{S}_f(w)\mathbf{R}_w(x)), \quad (2)$$

where $\mathbf{S}_f(w)$ is the Fourier spectrum for f, R_w are the group representations of orders r_w, K is the dimension of the dual object $\Gamma$for G, and $Tr(\mathbf{X})$ denotes the trace of a matrix $\mathbf{X}$.

In matrix notation, the Fourier transform pair defined by (1) and (2) can be **Error! Bookmark not defined.**expressed as follows [2]:

Given a function f on a group G of order g by the function vector $\mathbf{f}=[f(0),\ldots,f(g\text{-}1)]^T$. The Fourier spectrum for f , represented as a matrix-valued vector $[\mathbf{S}_f]=[\mathbf{S}_f(0),\ldots,\mathbf{S}_f(K\text{-}1)]^T$ is defined as

$$[\mathbf{S}_f] = g^{-1}[\mathbf{R}]^{-1} \bullet f, \quad (3)$$

where $[\mathbf{R}]^{-1} = [\mathbf{b}_{sq}]$, with $\mathbf{b}_{sq} = r_w \mathbf{R}_s^{-1}(q)$, $s=\{0,1,\ldots,K\text{-}1\}$, $q=\{0,1,\ldots,g\text{-}1\}$. The inverse transform is defined as

$$\mathbf{f} = [\mathbf{R}] \circ [\mathbf{S}_f] \quad (4)$$

where $\bullet$ and $\circ$ denote the generalized multiplications permitting dealing with vectors and matrices whose entries are matrices defined in [2], and $[\mathbf{R}]=[\mathbf{a}_{ij}]$, with $\mathbf{a}_{ij} = \mathbf{R}_j(i)$, $i=\{0,1,\ldots,g\text{-}1\}$, $j=\{0,1,\ldots,K\text{-}1\}$.

The matrix expression of the Fourier transform is important, since permits an elegant way towards devising the fast computing methods discussed in this paper.

III. Fast Algorithms for Computing the Fourier Transform

The fast Fourier transform (FFT) is a method for computing the Discrete Fourier transform (DFT) efficiently in terms of space and time [3].

This algorithm is based upon the factorization of the DFT transform matrix into the product of sparse matrices, each matrix describing a step in the FFT. The same approach can be used to devise fast algorithms for the Fourier transforms on finite groups, see for instance [2] and references therein. It is assume that the domain group G can be written as the direct product of groups $\mathbf{G}_i$, $i=\{0,1,\ldots,n\text{-}1\}$ of smaller orders. Then, the Fourier transform on G can be performed as n Fourier transforms on the constituent groups G_i. This could be considered as a restriction of the FFT from the whole group G to the FFT of its subgroups G_i. It follows that the i-th factor matrix can be represented as the Kronecker product of the Fourier transformation matrix on Gi of order gi at the i-th position and the identity matrices of orders gj, j $\in\{1,...,n\}\backslash\{i\}$,at all other positions into the Kronecker product. The same approach can be extended to non-Abelian groups using generalized matrix multiplications as described in [2].

The matrix [**R**] in the definition of the Fourier transform on finite non-Abelian groups is the matrix of unitary irreducible representations of G over P [2]. Matrix [**R**] and also its inverse $[\mathbf{R}]^{-1}$ can be generated as Kronecker products of ($\boldsymbol{K}_i \times g_i$) unitary irreducible representations of subgroups.

$$[\mathbf{R}] = \overset{n}{\underset{i=1}{\otimes}}[\mathbf{R}_i] \tag{6}$$

This matrix can be can be further factorized into elementwise Kronecker product of n sparse factors [Ci], i $\in$ {1,…,n} as

$$[\mathbf{C}^i] = \overset{n}{\underset{j=1}{\otimes}}[\mathbf{S}^i_j], i = 1,....,n \tag{7}$$

where

$$[\mathbf{S}^i_j] = \begin{cases} \mathbf{I}_{(g_j \times g_j)}, j < i \\ [\mathbf{R}_j]^{-1}, j = i \\ \mathbf{I}_{(K_j \times K_j)}, j > i \end{cases} \tag{8}$$

and $\mathbf{I}_{a\times a}$ is an ($a\times a$) identity matrix.

Although the (a) classical FFT algorithm can be derived in a similar fashion, there are some important differences with respect to dealing with finite non-Abelian group. Elements of vectors are not just scalars but could also be square matrices depending on (of) the representation of the corresponding non-Abelian group. Therefore, the number of representations of a non-Abelian group is always smaller than the order of that group. This implies that the vector size in different steps of FFT varies.

Fast algorithm is performed in n steps where n is (represents) the number of subgroups of G. The time complexity of each step is linear with the vector size.

IV. Implementation

We implemented the algorithm for computing the Fourier spectrum on finite non-necessarily Abelian groups using the C++ programming language. Since vector elements can be either scalars or matrices, we resorted to usage of OO techniques by abstracting the vector element as a class *Element [Complex]* on which we define the generalized multiplication operator as stated in the previous section.

Numbers of subgroups as well as the subgroups themselves for the each step could be chosen by the user, giving the chance to find the best suited decomposition for the problem at hand. Some of the subgroups implemented include: the additive groups of integers Z2, Z4, Z8, the symmetric group of permutations of order three S3, and the quaternion group Q8 [4]. This is implemented as a quad pointer to the object of the class element which represents the n-element vector of the subgroup transformation matrices with entries in the class *Element*. Since the size of vector during the different steps varies, we need to maintain the current vector size. In order to achieve the optimal space complexity in each step, memory is dynamically allocated and later de-allocated. The products of sizes of the left and the right identity matrices $\mathbf{S}^i_j$ defined by (8) are also maintained for fast multiplication of sparse matrices.

The algorithm for computing the Fourier spectrum can be specified in the pseudo code as:

```
FFT(N,g,K,V,NV)
1.  l ← 1
2.  for i ← 1 to N-1
3.    l ← l*g[i]
4.  d ← 1
5.  NX ← NV
6.  X ← V
7.  for i1 ← 0 to N-1
8.    mat ← mats[i1]
9.    NY ← NX*K[i1]/g[i1]
10.   KX=NX/l
11.   KY=NY/l
12.   for i2 ← 0 to l-1
13.     for i3 ← 0 to d-1
14.       for I ← 0 to K[i1]-1
15.         Y[i2*KY+i*d+i3] ← 0
16.         for j ← 0 to g[i1]-1
17.           Y[i2*KY+i*d+i3]←mat[i][j]* X[i2*KX+j*d+i3]
        +  Y[i2*KY+i*d+i3]
18.     l←l/g[i1+1]
19.     d←d*K[1i]
20.     X ← Y
```

21. NX ← NY

In the algorithm above *N* represent the number of steps i.e. the number of subgroups G is factored into, g and K represent arrays of sizes of those subgroups and their dual objects [2] respectively, while V is the input vector and NV its size. As mentioned above, sizes of left and right identity matrices are represented with *l* and *d* which are set to $\prod_{i=1}^{n-1} g[i]$ and 1 respectively in lines 1-4. A temporary vector X of size NX is used for containment at various steps and is initially set to V and NV at lines 5-6. In lines 7-21, the Fourier spectrum is computed in *n* steps. In lines 8-9 the i^{th} transformation matrix is loaded and the corresponding size of the vector Y is set. Each step could be performed by dividing the input vector into *l* parts each consisting of d interlacing g[i1] × K[i1] butterflies.

In lines 12-17, the algorithm iterates through the parts and then trough butterflies performing the i^{th} transformation on the appropriate vector elements. In lines 18-19, new values for *l* and *d* are calculated.

V. Experimental Results

Asymptotic time complexity of FFT is $O(n\log n)$ where *n* is the size of the input vector. Since the complexity of FFT over vectors does not depend on the values f vector entries, the computations are performed over randomly generated integer vectors of different sizes.

In the following table we present the running time of algorithm depending on the input size when the domain group *G* is factored in the product of subgroups Z_2.

TABLE I
Running time comparison with same factoring and different vector sizes.

Input vector size	Running time [ms]
32	2
128	15
256	16
512	31
1024	31
2048	93
4096	203
8192	468
16384	1061

Next we present the running time of algorithm depending on the input size when factoring the froup to the subgroups Q8

TABLE II
Running time comparison with same factoring and different vector sizes.

Input vector size	Running time [ms]
8	0
64	1
512	31
4096	156
32768	1314
262144	12324

The same results will be shown graphically it the figures 1 and 2 for better running time comparison of factoring same vectors over different subgroups.

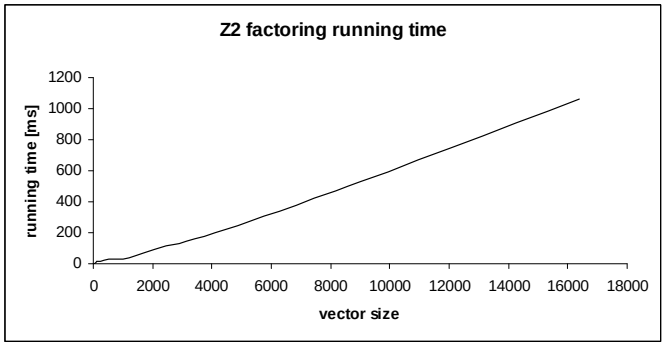

Fig.1. Z2 factoring running time.

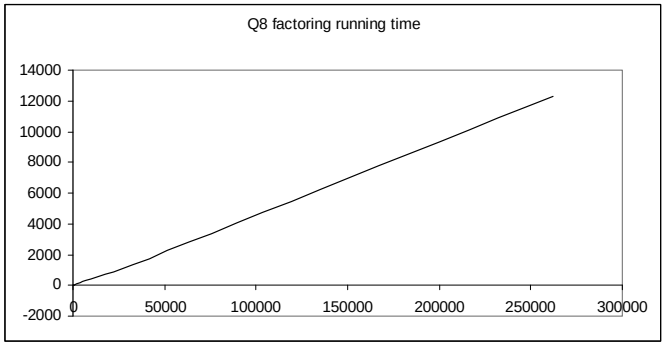

Fig.2. Q8 factoring running time.

As shown in the Figures 1 and 2, with our non-Abelian algorithm the asymptotic complexity remains the same, but with proper group choice, number of operations can be dramatically reduced.

We present the obtained results of running time over the same vector of size 214 while using different factoring.

TABLE III
Running time comparison with different factoring and same vector sizes.

Factoring	Running time [ms]
Z_2^{14}	1061
$Z_8^4 \times Z_2^2$	1029
Z_4^7	780
$Q_8^2 \times Z_8^2 \times Z_4$	718
$Q_8^4 \times Z_2^2$	624

Apart from the running time, different factoring also affects the sparseness of the function spectra.

We present the obtained results of non-zero elements over the same vector of size 2^5 while using different factoring.

TABLE IV
COMPARISON OF NON-ZERO ELEMENTS OVER DIFFERENT FACTORING AND SAME VECTOR.

Factoring	Percentage of non-zero elements
$Q_8 \times Z_2^2$	90%
$Z_2 \times Z_8$	53.13%
$Z_8 \times Z_4$	31.20%
$Z_2 \times Z_4^2$	15.62%
$Z_4 \times Z_2^3$	9.37%
Z_2^5	6.25%

VI. CONCLUSION

Group theoretical approach provides us with uniform treatment of signals defined over different algebraic structures. The first part of our work was to implement FFT when the user can arbitrary select factorization into subgroups. The second part was devoted to experimentally analyze how different factoring affects output data and time complexity of the algorithm.

As shown above, by a careful choice of subgroups we can obtain better running time and spectra sparseness. Apart of that different group choices can provide us with various perspectives of function properties.

The implementation can be further improved by adding more group choices and by code optimization of calculations at each step.

REFERENCES

[1] W. Rudin, *Fourier Analysis on Groups*, New York, Interscience Publishers, 1962.

[2] R. S. Stankovic, C. Moraga, J.T. Astola, *Applications of Fourier Analysis on Finite non-Abelian Groups in Signal Processing and System Design*, New York, John Wiley & Sons, 2005.

[3] C. V. Loan, *Computational Frameworks for the Fast Fourier Transform*, Society for Industrial Mathematics, 1992.

[4] Rober B. Ash, *Basic Abstract Algebra: For Graduate Students and Advanced Undergraduates,* Dover books on Mathematics, 2000.