9.1 Introduction

From signal processing point of view, the derivative for functions of real variables, which is a notion originated by Newton and converted into a strong mathematical concept by Leibniz, can be viewed as an operator intended to estimate the rate of change and the direction of change of a signal for the infinitesimally small change of its argument. The same idea of having such an operator in the context of Walsh analysis can be mentioned as a motivation for the introduction of the Gibbs derivative. Being an operator for functions defined on dyadic group, the shift of the argument is defined in terms of the componentwise addition modulo 2 of binary representations of the argument and the increment. A thorough analysis of relationships, similarities and differences, between the Newton-Leibniz and Gibbs derivatives is presented in [5].

For the application of a mathematical operator in engineering practice it is often required to provide efficient computation methods. In this chapter, we discuss fast algorithms to compute Gibbs derivatives on finite Abelian groups and their implementation on graphics processing units (GPUs).

Already in the initial publications about the concept, the Gibbs dyadic derivative on finite dyadic groups was formulated as a convolution operator [3]. This immediately leads to fast computation algorithms by using the convolution theorem and relationships of the Gibbs derivative and Fourier transform on finite groups. The involved computations are performed by using the Fast Walsh (Fourier) transform (FFT).
Alternatively, the Gibbs dyadic derivative can be defined as a linear combination of partial Gibbs dyadic derivatives resembling the definition of the Boolean difference [12]. In this case, it can be computed by FFT-like algorithm performing separate steps of the FFT on finite dyadic groups to the initial function \( f \) to be processed [13].

Both definitions of the Gibbs dyadic derivative on finite dyadic groups can be directly generalized to the Gibbs derivatives on finite Abelian groups by replacing the discrete Walsh functions (group characters of the finite dyadic groups) with group characters of the corresponding groups [11]. The same computing methods as for the Gibbs dyadic derivative can be used also in the case of Gibbs derivatives on finite Abelian groups [12], [15]. These methods are based on steps of FFT-like algorithms, and therefore they are very suitable for the implementation on GPUs, as will be documented by experimental results presented in this chapter.

### 9.2 Definitions of Gibbs Derivatives on Finite Abelian Groups

Initially, the Gibbs derivative was defined on the finite dyadic group \( C_2^n \) consisting of the set of binary \( n \)-tuples (\( n \) is a natural number) \( X = (x_0, x_1, \ldots, x_{n-1}) \) with \( x_i \in \{0, 1\} \) equipped with the componentwise addition modulo 2.

With each element of \( C_2^n \) it can be associated a unique element of the set of non-negative integers less than \( 2^n \), \( B_n = \{0, 1, \ldots, 2^n - 1\} \) by means of a function \( V_n : C_2^n \to B_n \) defined by

\[
V_n(x) = \sum_{i=0}^{n-1} 2^{n-i-1}x_i.
\]

The space of all bounded complex-valued functions \( f \) on \( C_2^n \) (or on \( B_n \)) will be denoted by \( L_n \). With this notation, the Gibbs dyadic derivative is defined as follows [3].

**Definition 9.1** To each complex-valued function \( f \) on \( C_2^n \) we assign a function \( f^{[1]} \in L_n \) defined by

\[
f^{[1]}(x) = -\frac{1}{2} \sum_{r=0}^{n-1} (f(x \oplus 2^r) - f(x))2^r, \quad (x \in C_2^n).
\]

In matrix notation, the Gibbs derivative on finite dyadic groups is defined as

**Definition 9.2** The dyadic Gibbs derivative is defined by a \( (2^n \times 2^n) \) matrix \( D = [d_{i,j}] \) whose entries are

\[
d_{\xi,\eta} = \frac{1}{2}
\left((2^n - 1)\delta(\xi \oplus \eta, 0) - \sum_{r=0}^{n-1} 2^r \delta(\xi \oplus \eta, 2^r)\right).
\]