# GFDM-based Multicarrier Waveform Generator Tutorial

Ahmad Nimr
Vodafone Chair Mobile Communication Systems, Technische Universität Dresden, Germany
{ahmad.nimr}@ifn.et.tu-dresden.de

## I. Multicarrier waveforms overview

In general multicarrier modulation, a stream of data symbols is split into time-frequency sub streams $d_{k,m,i}$, with $k$ is the index in the frequency domain known as subcarrier, $m$ the index in the time domain known as subsymbol and $i$ represents the block index. Each stream is modulated with transmitter pulse shapes $g^t_{k,m}[n]$, and the discrete transmitted signal can be written as

$$\begin{aligned} x^t[n] &= \sum_{i=-\infty}^{\infty} \sum_{k\in K_{\text{on}}} \sum_{m\in M_{\text{on}}} d_{k,m,i} g^t_{k,m}[n - iN_s] \\ &= \sum_{i=-\infty}^{\infty} x^t_i[n - iN_s]. \end{aligned} \tag{1}$$

where $N_s$ is the symbol spacing, $\mathcal{K}_{\text{on}}$ and $\mathcal{M}_{\text{on}}$ are the sets of active subcarriers and subsymbols, respectively, and $x^t_i[n]$ is the $i$-th multicarrier symbol, which is given by

$$x^t_i[n] = \sum_{k\in K_{\text{on}}} \sum_{m\in M_{\text{on}}} d_{k,m,i} g^t_{k,m}[n]. \tag{2}$$

In the common modulation techniques $g^t_{k,m}[n]$ has finite length $N_t$ and can be generated using prototype pulse shape $g[n]$ with circular shift in time and frequency such that

$$\begin{aligned} g^t_{k,m}[n] &= g[< n - mK - n_0 >_N] e^{j2\pi \frac{k}{K} n}, \\ n &= 0, \cdots, N_t - 1, \end{aligned} \tag{3}$$

where $K$ is the number of subcarriers, $M$ the number of subsymbols $N = MK \leq N_t$. Let $N_o$ be the overhead that represents cyclic prefix (CP) and/or cyclic suffix (CS), where $N_t = N + N_o$. The type of overhead is defined by the time shift $n_0$. For instance, when $n_0 = N_o$ we get pure CP. In this context, adding CP enables low complexity frequency fomain equalization (FDE). Based on that, we can define a core block $x_i[n]$ of length $N$ samples such that

$$x_i[n] = \sum_{k\in K_{\text{on}}} \sum_{m\in M_{\text{on}}} d_{k,m,i} g[< n - mK >_N] e^{j2\pi \frac{k}{K} n}. \tag{4}$$

The relation between the core block and the actual transmitted block is given by

$$x^t_i[n] = x_i[< n - n_0 >_N]. \tag{5}$$

Furthermore, two specific cases evolve depending on $N_s$ and $N_t$

- $N_s \geq N_t$: in this case the transmitted blocks $x^t_i[n]$ do not overlap and the total transmitted signal $x^t[n]$ can be generated by transmitting $x^t_i[n]$ independently. Here, $N_P = N_s - N_t$ represents a guard interval between sequential blocks. The block based waveforms such as orthogonal frequency division multiplexing (OFDM) and generalized frequency division multiplexing (GFDM) belong to this case.
- $N_s < N_t$: $x^t[n]$ can be still produced based on $x^t_i[n]$, however, additional overlapping of length $N_o = N_t - N_s$ must be considered. In this, the last $N_o$ samples of the current $x^t[n]$ added to the first $N_o$ samples of the next $x^t_i[n]$. An example of this case is filtered multitone (FMT) with overlapping factor $M$. In witch, $\mathcal{M}_{\text{on}}$ contains only one subsymbols and the remaining $M - 1$ are non-active. However the overlapping by $N_o = K(M - 1)$ compensate this overhead. Another example, where the ramp-down of the $i - 1$-th windowed block overlaps with ramp-up of the $i$-th block.

As a result, by the configuration of the prototype pulse shape $g$ in addition to the parameters, $N_s$, $N_t$, $n_0$, $K$, $M$, $\mathcal{K}_{\text{on}}$ and $\mathcal{M}_{\text{on}}$ we can produce most of the state of the art waveforms.

As illustrated in Fig. 1, the process of waveform generation can be divided into two main stages; the first is computation of the core block $x_i[n]$ according to (4). The second stage corresponds to block multiplexing where the CP/CS overhead is added

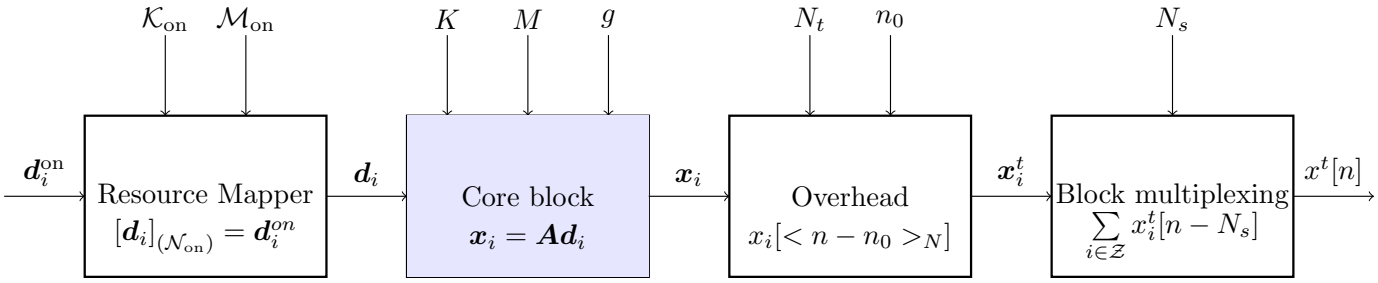


Fig. 1: Multicarrier waveforms generator stages.

to generate $x_i^t[n]$ as done in (2). At this stage, further signal processing items can be performed afterwards, such as windowing or filtering. Afterwards, the transmitted signal is generated according to (1) considering overlapping or zero padding. Finally, the discrete block is converted to analogue signal using digital-to-analogue converter (DAC) with sampling rate $F_s$. Therefore, the subcarrier spacing can be derived as $\Delta f = \frac{F_s}{K}$. and the signal bandwidth can be controlled via $\mathcal{K}_{\text{on}}$.

## II. Core block representation

Next we focus on the computation of (4). For notation simplicity, we drop the block index $i$ and represent the core block in a vector $\boldsymbol{x} \in \mathbb{C}^{N\times 1}$, so that

$$[\boldsymbol{x}]_{(n)} = x(\tfrac{n}{B}) = \sum_{k=0}^{K-1}\sum_{m=0}^{M-1} d_{k,m} g[< n - mK >_N] e^{j2\pi\frac{k}{K}n}. \tag{6}$$

The corresponding representation in the frequency domain is achieved using $\tilde{x} = N\text{-DFT}\{\boldsymbol{x}\}$ and given by

$$[\tilde{\boldsymbol{x}}]_{(n)} = \sum_{k=0}^{K-1}\sum_{m=0}^{M-1} d_{k,m} \tilde{g}[< n - kM >_N] e^{-j2\pi\frac{m}{M}n}. \tag{7}$$

The GFDM demodulator applies circular filtering on the received signal $y[n]$ with a prototype receiver pulse $\boldsymbol{\gamma}[n]$, such that

$$\hat{d}_{k,m} = \sum_{n=0}^{N-1} y[n]\gamma^*[< n - mK >_N] e^{-j2\pi\frac{k}{K}n}. \tag{8}$$

In the conventional matrix representation, the GFDM block can be expressed using a matrix $\boldsymbol{A} \in \mathbb{C}^{N\times N}$ as

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{d}, [\boldsymbol{A}]_{(n,k+mK)} = g[< n - mK >_N] e^{j2\pi\frac{k}{K}n}, \tag{9}$$

where $\boldsymbol{d} = \text{vec}\{\boldsymbol{D}\}$ with $[\boldsymbol{D}]_{(k,m)} = d_{k,m}$. Moreover, the demodulation matrix $\boldsymbol{B} \in \mathbb{C}^{N\times N}$ has a similar structure and is generated from the receiver pulse $\gamma$ as

$$[\boldsymbol{B}]_{(n,k+mK)} = \gamma[< n - mK >_N] e^{j2\pi\frac{k}{K}n},\ \hat{\boldsymbol{d}} = \boldsymbol{B}^H\boldsymbol{y}. \tag{10}$$

### A. Alternative representation

In this section, we introduce an alternative matrix model by arranging the GFDM block $\boldsymbol{x}$ in a matrix instead of a vector. Initially, we define the following auxiliary matrices for a generic vector $\boldsymbol{a} \in \mathbb{C}^{PQ\times 1}$ as visualized in Fig. 2,

$$\boldsymbol{V}_{P,Q}^{(\boldsymbol{a})} = (\text{unvec}_{Q\times P}\{\boldsymbol{a}\})^T \Leftrightarrow \left[\boldsymbol{V}_{P,Q}^{(\boldsymbol{a})}\right]_{(p,q)} = [\boldsymbol{a}]_{(q+pQ)}, \tag{11}$$

$$\boldsymbol{Z}_{P,Q}^{(\boldsymbol{a})} = \boldsymbol{F}_P \boldsymbol{V}_{P,Q}^{(\boldsymbol{a})},\ \bar{\boldsymbol{Z}}_{Q,P}^{(\tilde{\boldsymbol{a}})} = \frac{1}{Q}\boldsymbol{F}_Q^H \boldsymbol{V}_{Q,P}^{(\tilde{\boldsymbol{a}})}, \tag{12}$$

where $\tilde{\boldsymbol{a}} = \boldsymbol{F}_{PQ}\boldsymbol{a}$, $\text{unvec}_{Q\times P}\{\boldsymbol{a}\}$ denotes the inverse of vectorization operation, $\boldsymbol{F}_P$ is the $P$-point discrete Fourier transform (DFT) matrix, defined by $[\boldsymbol{F}_P]_{(i,j)} = e^{-j2\pi\frac{ij}{P}}$. The matrix $\boldsymbol{Z}_{P,Q}^{(\boldsymbol{a})}$, $\bar{\boldsymbol{Z}}_{Q,P}^{(\tilde{\boldsymbol{a}})}$ are known as the discrete Zak transform (DZT) [**?**] of $\boldsymbol{a}$ and $\tilde{\boldsymbol{a}}$, respectively.

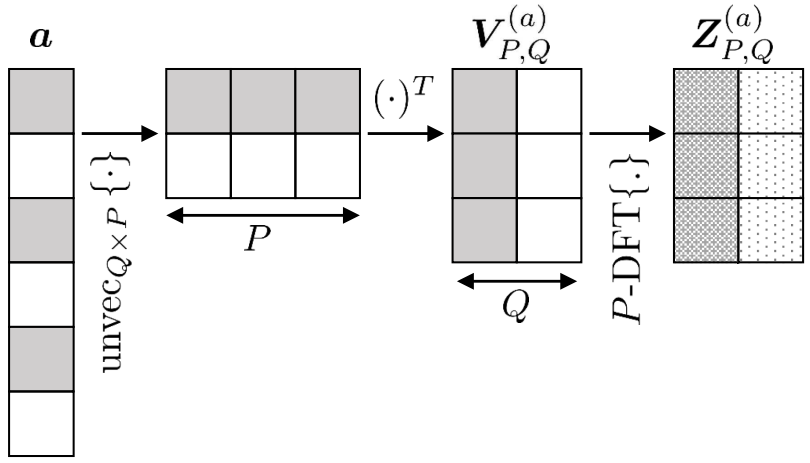


Fig. 2: Visualization of (11) and (12).

*1) Time domain representation:* The GFDM block equation (6) can be reformulated by representing $n$ with two indexes $q = 0, \cdots, K-1$ and $p = 0 \cdots, M-1$, such that $n = q + pK$. Thereby,

$$[\boldsymbol{x}]_{(q+pK)} = \sum_{m=0}^{M-1} \sum_{k=0}^{K-1} d_{k,m} g[< q + pK - mK >_N] e^{j2\pi \frac{k}{K} q}$$

Using the notations in (11), then

$$\begin{aligned}\left[\boldsymbol{V}_{M,K}^{(\boldsymbol{x})}\right]_{(p,q)} &= \sum_{m=0}^{M-1} \left[\boldsymbol{V}_{M,K}^{(\boldsymbol{g})}\right]_{(<p-m>_M,q)} \sum_{k=0}^{K-1} [\boldsymbol{D}]_{(k,m)} e^{j2\pi \frac{k}{K} q} \\ &= \sum_{m=0}^{M-1} \left[\boldsymbol{V}_{M,K}^{(\boldsymbol{g})}\right]_{(<p-m>_M,q)} \left[\boldsymbol{D}^T \boldsymbol{F}_K^H\right]_{(m,q)}.\end{aligned}$$

The second line defines circular convolution between the $q$-th column of $\boldsymbol{V}_{M,K}^{(\boldsymbol{g})}$ and the $q$-th column of $\boldsymbol{D}^T \boldsymbol{F}_K^H$, which can be then expressed in the frequency domain with $M$-DFT as

$$\left[\boldsymbol{F}_M \boldsymbol{V}_{M,K}^{(\boldsymbol{x})}\right]_{(:,q)} = \left[\boldsymbol{F}_M \boldsymbol{V}_{M,K}^{(\boldsymbol{g})}\right]_{(:,q)} \odot \left[\boldsymbol{F}_M \boldsymbol{D}^T \boldsymbol{F}_K^H\right]_{(:,q)}. \tag{13}$$

By stacking the columns according to the $q$-th index and using (12), we get

$$\boldsymbol{V}_{M,K}^{(\boldsymbol{x})} = \frac{1}{MK} \boldsymbol{F}_M^H \left(K \boldsymbol{Z}_{M,K}^{(\boldsymbol{g})} \odot \left[\boldsymbol{F}_M \boldsymbol{D}^T \boldsymbol{F}_K^H\right]\right). \tag{14}$$

Here, $\odot$ denotes the element-wise multiplication operator.

*2) Frequency domain representation:* Following similar derivation on $\tilde{\boldsymbol{x}}$ defined in (7), we get

$$\left[\boldsymbol{V}_{K,M}^{(\tilde{\boldsymbol{x}})}\right]_{(q,p)} = \sum_{k=0}^{K-1} \left[\boldsymbol{V}_{K,M}^{(\tilde{\boldsymbol{g}})}\right]_{(<q-k>_K,p)} \sum_{m=0}^{M-1} d_{k,m} e^{-j2\pi \frac{m}{M} p}.$$

and therefore,

$$\boldsymbol{V}_{K,M}^{(\tilde{\boldsymbol{x}})} = \frac{1}{K} \boldsymbol{F}_K \left(K \bar{\boldsymbol{Z}}_{K,M}^{(\tilde{\boldsymbol{g}})} \odot \left[\boldsymbol{F}_K^H \boldsymbol{D} \boldsymbol{F}_M\right]\right), \tag{15}$$

### *B. Four steps representation*

Based on the representations in the time domain (14) and the frequency domain (15), the GFDM modulation can be split into four steps, as shown in Fig. 3:

*1) Data spreading :* the spreading is achieved by applying DFT on the rows and inverse discrete Fourier transform (IDFT) on the columns of $\boldsymbol{D}$. Therefore, we get the spread data matrix

$$\boldsymbol{D}_s = \frac{1}{K} \boldsymbol{F}_K^H \boldsymbol{D} \boldsymbol{F}_M. \tag{16}$$

*2) Windowing:* the spread data matrix $\boldsymbol{D}_s$ is element-wise multiplied with a transmitter windowing matrix $\boldsymbol{W}_{\text{tx}} \in \mathbb{C}^{K\times M}$, which is generated based on the prototype pulse shape,

$$\boldsymbol{X} = \boldsymbol{W}_{\text{tx}} \odot \boldsymbol{D}_s. \tag{17}$$

the entries of $\boldsymbol{W}_{\text{tx}}$ depends on the implementation domain,

$$\boldsymbol{W}_{\text{tx}}^{(\text{TD})} = K\boldsymbol{Z}_{M,K}^{(\boldsymbol{g})^{T}}, \boldsymbol{W}_{\text{tx}}^{(\text{FD})} = K\boldsymbol{Z}_{K,M}^{(\tilde{\boldsymbol{g}})}. \tag{18}$$

*3) Transformation:* the matrix $\boldsymbol{X}$ can be seen as frequency domain blocks of length $M$ in rows, or as time domain symbols of length $K$ in columns. Thus, a final $M$-DFT or $K$-IDFT is applied to obtain the block samples, as follows

$$\boldsymbol{V}_{M,K}^{(\boldsymbol{x})} = \tfrac{1}{M}\boldsymbol{F}_M^H\boldsymbol{X}^T, \quad \boldsymbol{V}_{K,M}^{(\tilde{\boldsymbol{x}})} = \boldsymbol{F}_K\boldsymbol{X} \ . \tag{19}$$

*4) Allocation:* the final vector is achieved by allocating the generated samples to the corresponding indexes. Specifically,

$$\boldsymbol{x} = \text{vec}\left\{\left(\boldsymbol{V}_{M,K}^{(\boldsymbol{x})}\right)^T\right\}, \quad \tilde{\boldsymbol{x}} = \text{vec}\left\{\left(\boldsymbol{V}_{K,M}^{(\tilde{\boldsymbol{x}})}\right)^T\right\}. \tag{20}$$

Additionally, $N$-IDFT is required for the frequency domain implementation to get the time domain signal, i.e.

$$\boldsymbol{x} = \frac{1}{N}\boldsymbol{F}_N\tilde{\boldsymbol{x}}. \tag{21}$$

The demodulator Fig. 4 performs the inverse steps. First, the matrix $\boldsymbol{Y} \in \mathbb{C}^{K\times M}$ is constructed from the received signal $\boldsymbol{y}_{\text{eq}}$,

$$\boldsymbol{Y}^{(\text{TD})} = \left[\boldsymbol{F}_M\boldsymbol{V}_{M,K}^{(\boldsymbol{y}_{\text{eq}})}\right]^T, \quad \boldsymbol{Y}^{(\text{FD})} = \tfrac{1}{K}\boldsymbol{F}_K^H\boldsymbol{V}_{K,M}^{(\tilde{\boldsymbol{y}}_{\text{eq}})}. \tag{22}$$

Then a receive window $\boldsymbol{W}_{\text{rx}}$ is applied to $\boldsymbol{Y}$ followed by despreading, so that

$$\hat{\boldsymbol{D}} = \frac{1}{M}\boldsymbol{F}_K\left(\boldsymbol{W}_{\text{rx}} \odot \boldsymbol{Y}\right)\boldsymbol{F}_M^H. \tag{23}$$

Equation (23) is an alternative representation of the circular demodulation in (8). Table I summarizes all the processing steps.

### *C. Extended flexibility*

In addition to the main configuration parameters $K$, $M$ and the prototype pulse shape, further degrees of freedom can be exploited to extend the flexibility of the modem, namely:

- Flexible spreading: the spreading can be altered to enable or disable the spreading matrices.
- Flexible transformation: the DFT or IDFT transformation can be turned on or off in both domains to get more options. This extended flexibility is exploited in the next section to generate the Orthogonal time frequency space modulation (OTFS) signal.
- Flexible allocation: the conventional allocation preserves the spectral characteristics of the GFDM signal. However, the mapping can be customized by changing the distribution of the samples in the transmitted signal.
- Flexible overhead: the overhead block can be redesigned to insert CPs within the core block.

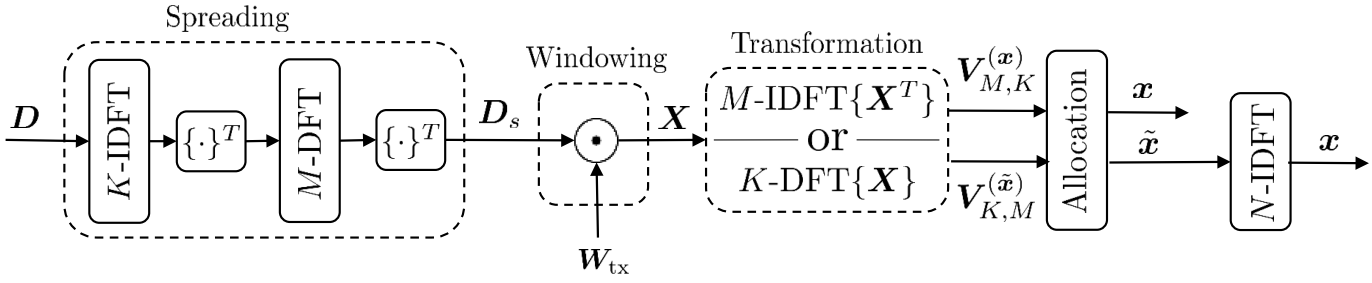


Fig. 3: GFDM-based modulator in 4 steps

In summary, the samples of the core block are represented in the matrix $\boldsymbol{V}_{M,K}^{(\boldsymbol{x})}$ or $\boldsymbol{V}_{M,K}^{(\tilde{\boldsymbol{x}})}$ which are generated from the same architecture, which is the crucial computation. The allocation and the overhead can be integrate in one block, this block takes $N$ samples and produces $N_t$ samples.

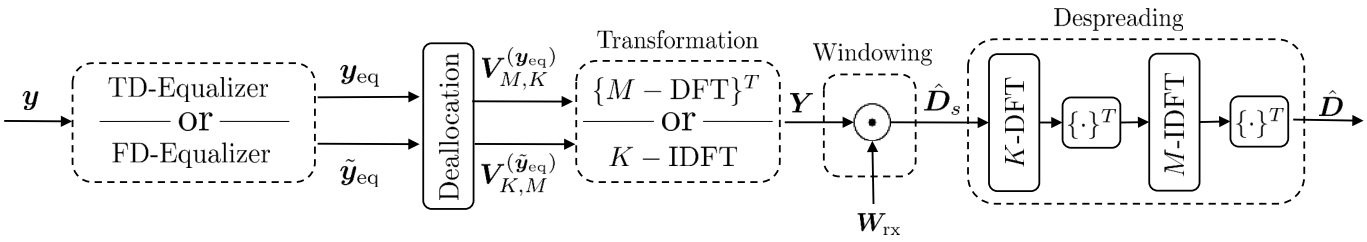


Fig. 4: GFDM-based demodulator.

TABLE I: The four steps of conventional GFDM modulator.

| | Modulation | | Demodulation | |
|---|---|---|---|---|
| | TD implementation | FD implementation | TD implementation | FD implementation |
| Spread. | $\boldsymbol{D}_s = \frac{1}{K}\boldsymbol{F}_K^H \boldsymbol{D}\boldsymbol{F}_M$ | | $\hat{\boldsymbol{D}} = \frac{1}{M}\boldsymbol{F}_K \hat{\boldsymbol{D}}_s \boldsymbol{F}_M^H$ | |
| Wind. | $\boldsymbol{W}_{\text{tx}} = K\{\boldsymbol{Z}_{M,K}^{(\boldsymbol{g})}\}^T$ | $\boldsymbol{W}_{\text{tx}} = K\bar{\boldsymbol{Z}}_{K,M}^{(\tilde{\boldsymbol{g}})}$ | $\boldsymbol{W}_{\text{rx}}$ generated from $\boldsymbol{W}_{\text{tx}}$ | |
| | $\boldsymbol{X} = \boldsymbol{W}_{\text{tx}} \odot \boldsymbol{D}_s$ | | $\hat{\boldsymbol{D}}_s = \boldsymbol{W}_{\text{rx}} \odot \boldsymbol{Y}$ | |
| Trans. | $\boldsymbol{V}_{M,K}^{(\boldsymbol{x})} = \frac{1}{M}\boldsymbol{F}_M^H \boldsymbol{X}^T$ | $\boldsymbol{V}_{K,M}^{(\tilde{\boldsymbol{x}})} = \boldsymbol{F}_K \boldsymbol{X}$ | $\boldsymbol{Y} = \left[\boldsymbol{F}_M \boldsymbol{V}_{M,K}^{(\boldsymbol{y}_{\text{eq}})}\right]^T$ | $\boldsymbol{Y} = \frac{1}{K}\boldsymbol{F}_K^H \boldsymbol{V}_{K,M}^{(\tilde{\boldsymbol{y}}_{\text{eq}})}$ |
| Alloc. | $\boldsymbol{x} = \text{vec}\left\{\{\boldsymbol{V}_{M,K}^{(\boldsymbol{x})}\}^T\right\}$ | $\tilde{\boldsymbol{x}} = \text{vec}\left\{\{\boldsymbol{V}_{K,M}^{(\tilde{\boldsymbol{x}})}\}^T\right\}$ | $\left[\boldsymbol{V}_{M,K}^{(\boldsymbol{y}_{\text{eq}})}\right]_{(m,k)} = [\boldsymbol{y}_{\text{eq}}]_{(k+mK)}$ | $\left[\boldsymbol{V}_{K,M}^{(\tilde{\boldsymbol{y}}_{\text{eq}})}\right]_{(k,m)} = [\tilde{\boldsymbol{y}}_{\text{eq}}]_{(m+kM)}$ |

## III. MATLAB REFERENCE IMPLEMENTATION

Consider a multicarrier system with the parameters $K$, $M$, $\boldsymbol{g}$, $\mathcal{K}_{\text{on}}$, $\mathcal{M}_{\text{on}}$. The resource mapping is implemented in the functions

```
function D = ResourceMap(d, K_set, M_set, K,M)
function d = ResourceDemap(D, K_set, M_set)
```

| Parameter | Notation | description |
|---|---|---|
| d | $\boldsymbol{d}$ | input data symbols of size $\lvert\mathcal{K}_{\text{on}}\rvert\lvert\mathcal{M}_{\text{on}}\rvert \times 1$ |
| D | $\boldsymbol{D}$ | data symbols of size $K \times M$. It contains zeros at the non allocated positions |
| K_set | $< \mathcal{K}_{\text{on}} >_K +1$ | active set of subcarriers given according to Matlab indexing. |
| M_set | $< \mathcal{M}_{\text{on}} >_M +1$ | active set of subcarriers given according to Matlab indexing. |
| K | $K$ | number of subcarriers |
| M | $M$ | number of subsymbols. |

The first three steps in Table I are realized using fast Fourier transform (FFT) and element-wise multiplication. The function *GeneralModulator* is used for modulation

```
function X = GeneralModulator( D, K_set, W, type )
```

| Parameter | Notation | description |
|---|---|---|
| D | $\boldsymbol{D}$ | input data symbols of size $K \times M$ |
| K_set | $< \mathcal{K}_{\text{on}} >_K +1$ | active set of subcarriers given according to Matlab indexing. This is used to reduce the number of $M$-FFT operations. |
| W | $\boldsymbol{W}_{tx}$ | Transmitter window of size $K \times M$. The function *PulseToWindow* is used to generate it. |
| type | - | Type of implementation it can be *'TD' or 'FD'*. |
| X | $\boldsymbol{V}_{M,K}^{(\boldsymbol{x})}$, or $\boldsymbol{V}_{K,M}^{(\tilde{\boldsymbol{x}})}$ | output samples, either $M \times K$ for *'TD'* mode or $K \times M$ for *'FD'* mode |

To generate the transmit window, the function *PulseToWindow* is used.

```
function Wt = PulseToWindow( g, K, M, type)
```

| Parameter | Notation | description |
|---|---|---|
| g | $\boldsymbol{g}$ | input pulse shape of size $KM \times 1$ either in time or frequency domain depending on the modulator type. |
| K | $K$ | number of subcarriers |
| M | $M$ | number of subsymbols. |
| type | - | Type of implementation it can be *'TD' or 'FD'*. |
| Wt | $\boldsymbol{W}_{\text{tx}}$ | transmitter window of size $K \times M$, it can be scalar also. |

The revere operation to get a pulse from window is implemented in *WindowToPulse*.

```
function g = WindowToPulse( W, type)
```

The demodulator is implemented in the function *GeneralDemodulator*

```
function D = GeneralDemodulator( X, K_set, W, type )
```

| Parameter | Notation | description |
|---|---|---|
| X | $\boldsymbol{V}_{M,K}^{(\boldsymbol{y}_{\text{eq}})}$, or $\boldsymbol{V}_{K,M}^{(\tilde{\boldsymbol{y}}_{\text{eq}})}$ | equalized samples, either $M \times K$ for *'TD'* mode or $K \times M$ for *'FD'* mode |
| K_set | $< \mathcal{K}_{\text{on}} >_K +1$ | active set of subcarriers given according to Matlab indexing. This is used to reduce the number of $M$-FFT operations. |
| W | $\boldsymbol{W}_{\text{rx}}$ | receiver window of size $K \times M$. The function *ReceiveWindow* is used to compute the receive window. |
| type | - | Type of implementation it can be *'TD' or 'FD'*. It can be scalar also. |
| D | $\hat{\boldsymbol{D}}$ | output data symbols of size $K \times M$ |

The receive window is computed from the transmitter window using *ReceiveWindow* considering the model,

$$\boldsymbol{Y} = \boldsymbol{W}_{\text{tx}} \odot \boldsymbol{D}_s + \boldsymbol{V}, \tag{24}$$

where $\boldsymbol{V}$ is the additive noise, and generated from the noise vector $\boldsymbol{v}$ similar to $\boldsymbol{Y}$. The SNR matrix is devined by

$$[\text{SNR}]_{(k,m)} = \frac{\mathbb{E}\left[\,|\,[\boldsymbol{D}_s]_{(k,m)}\right]\,|^2}{\mathbb{E}\left[\,|\,[\boldsymbol{V}]_{(k,m)}\right]\,|^2}. \tag{25}$$

when $\{[\boldsymbol{D}_s]_{(k,m)}\}$ are uncorrelated, which is the case of full allocation and incorporated samples $d_{k,m}$, this model can be used to compute minimum mean square error (MMSE) window. The zero-forcing (ZF) window is computed by the element-wise division and the matched filer by complex conjugate.

```
function Wr = ReceiveWindow( Wt, rxType, SNR)
```

| Parameter | Notation | description |
|---|---|---|
| Wt | $\boldsymbol{W}_{\text{tx}}$ | transmitter window of size $K \times M$. |
| rxType | - | Type of receiver it can be *'MF', 'ZF' or 'MMSE'*. |
| SNR | Eq. (25) | A matrix of size $K \times M$. It is mainly used with *'MMSE'* type. It can be scalar also. |
| Wr | $\boldsymbol{W}_{\text{rx}}$ | receiver window of size $K \times M$ |

The allocation and CP insertion can pe performed depending on the target wave form. For example in conventional GFDM

```
x_gfdm = reshape(X.',N,1);
x_gfdm_CP = AddCpCs( x_gfdm, N, Ncp, 0);
```

in OTFS

```
X_otfs_cp = AddCpCs( X, M, Ncp, 0);
x_otfs =  X_otfs_cp(:);
```

The functions *AddCpCs* and *RemoveCpCs* are used to add and remove CP, CS.

```
function X = AddCpCs( X0, N, Ncp, Ncs)
function Xo = RemoveCpCs( X, N, n0);
```

| Parameter | Notation | description |
|---|---|---|
| X0 | $\boldsymbol{x}$ | Core block of size $N \times 1$ or a matrix of size $N \times N_B$, where $N_B$ is the number of blocks. |
| N | $N$ | Core block length. |
| Ncp | $N_{\text{cp}}$ | CP length |
| Ncs | $N_{\text{cs}}$ | CS length |
| n0 | $n_0$ | can be CP length or it could include sample delay |
| X | $\boldsymbol{x}^{\text{tx}}$ | block after adding CP and CS |