Time-Variant and Quasi-separable Systems[*] supplementary reading

.

- Fourier Techniques-

Klaus Diepold, Patrick Dewilde

Spring 2025

1 Introduction

Fourier Transformation is one of the most essential algorithmic tools if it comes to representing and processing signals. In this context we only consider time-discrete signals. Therefore, we focus on the Discrete Fourier Transformaton and its fast computational version, the Fast Fourier Transformation.

We start out with the process of computing the convolution product of two signals, which is usually encountered when determining the output signal of a discrete-time linear and time-invariant system as a response to a given input signal or sequence. Computing this convolution will lead us to the use of Toeplitz matrices for formulating the convolution as a matrix vector product.

A further step will bring us to the notion of the cyclic convolution and the corresponding cyclic Toeplitz matrix. The analysis of the eigenvalue decomposition of the cyclic Toeplitz matrix then will give rise to the Discrete Fourier Transformation (DFT). The Fast Fourier Transformation (FFT) is then an efficient way to compute the DFT by exploiting the structure inherent to the DFT matrix.

This write-up contains two major learnings:

1. The existence of the DFT is bound to the existence of a cyclic Toepltz matrix, such that the DFT represents the eigenvalue decomposition of the Toeplitz matrix.
2. The use of the DFT for computing output signals of linear discrete-time systems is limited to the family of time-invariant systems. Otherwise the cyclic Toeplitz matrix wouldn't appear.

[*]P. Dewilde, K. Diepold, A.-J. v.d. Veen. Time-Variant ans Quasi-separable Systems, Cambridge University Press, 2024

2 Linear Dynamical Systems

2.1 Standard Representation

We can use the equivalence between Linear Systems on the one hand and Linear Algebra on the other hand if we want to represent such systems or if we are computing with input and output signals of linear systems. In Figure 1 depicts a symbolic representation for an system, that takes input sequence $[u_k]$ and produces the output sequence $[y_k]$, which is determined as the transfer operator $\mathcal{T}\{\cdot\}$ applied to the input sequence.

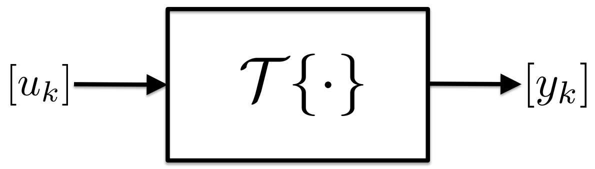

Figure 1: Input Output description of a linear time-invariant system.

Once we are dealing with *linear* systems, the sequences $[u_k]$ and $[y_k]$ are mapped into vectors u and y of length m and n,

$$[u_k] \mapsto u = \begin{bmatrix} \vdots \\ u_{k-1} \\ u_k \\ u_{k+1} \\ \vdots \end{bmatrix}, \qquad [y_k] \mapsto y = \begin{bmatrix} \vdots \\ y_{k-1} \\ y_k \\ y_{k+1} \\ \vdots \end{bmatrix},$$

respectively.

In digital signal processing and digital communications we are often interested in computing the output sequence $[y_k]$ of a linear time-invariant system, which is given in terms of the function $\mathcal{T}\{\cdot\}$, and which we have excited with an input sequence $[u_k]$. As shown in Figure 1, considering such a system, we observe that feeding the sequence $[u_k]$ to the input of the system causes the output sequence $[y_k]$, which is denoted as

$$[y_k] = \mathcal{T}\{[u_k]\},$$

where $[u_k] = \ldots u_{k-1}, u_k, u_{k+1} \ldots$, $[y_k] = \ldots y_{k-1}, y_k, y_{k+1} \ldots$ and $k = 1, 2, \ldots$ representing the time index for elements of the discrete-time sequence.

2.2 Linearity

We assume the systems to be linear such that we require that the superposition principle holds, that is, for two input sequences $[u_k]^1$ and $[u_k]^2$ the corresponding output sequences add like

$$[y_k]^1 = \mathcal{T}\{[u_k]^1\}, \quad [y_k]^2 = \mathcal{T}\{[u_k]^2\} \quad \Rightarrow \quad [y_k]^1 + [y_k]^2 = \mathcal{T}\{[u_k]^1 + [u_k]^2\}.$$

A consequence of the superposition principle reads as

$$\alpha \cdot [y_k] = \mathcal{T}\{\alpha \cdot [u_k]\},$$

where α is a scalar factor.

2.3 Time Invariance

The property of time-invariance states that the function $\mathcal{T}\{\cdot\}$ in invariant to shifts along the time axis, i.e. shifting the input sequence $[u_k]$ by τ causes a corresponding shift in the output sequence $[y_k]$

$$[y_{k-\tau}] = \mathcal{T}\{[u_{k-\tau}]\},$$

without causing further changes in $[y_k]$.

3 Exploiting Linearity and Time-Invariance

We can exploit the features of linearity and time-invariance to compute the output signal $[y_k]$ of the system. We show an example in Figure 2. We take an input sequence $[u_k]$ of length $m = 4$, and decompose it into the sum of individual impulses u_i, which are shifted in time. Each of these individual impulses generates a shifted version of the impulse response $[t_k]$ of length $n = 4$. Each of these shifted versions of the impulse response is weighted with the value of the corresponding input impulse u_i creating the individual impulse responses $[y_k]^i, i = 1, 2, 3, 4$. Here we exploit the time-variance property of the LTI-system such that the shifted versions of the impulse response are derived from the identical impulse response $[t_k]$. Finally, the output signal $[y_k]$ is generated as the sum of the individually weighted and shifted impulse responses. For this step we exploit the linearity property of the LTI-system (superposition principle).

Putting all this together we can observe that the LTI-system determines the output sequence $[y_k]$ as

$$[y_k] = \sum_{i=-\infty}^{\infty} [t_{k-i}] \cdot u_i, \quad k = 0, 1, \ldots m + n - 2,$$

which is called a *linear convolution.*

3.1 Input/Output representation using z-Transform

A popular tool for dealing with discrete time signals and systems is based on using the z-transformation of a time-series t_k, which is defined as

$$T(z) = \sum_{k=-\infty}^{\infty} t_k z^k. \tag{1}$$

The symbol z denotes a complex variable which turns a sequence of numbers t_k into a complex-valued function $T(z)$. Please note that here we use a positive exponent for z in our definition of the z-transformation. This is a minor modification in comparison to most standard engineering text books. However, in the mathematical literature the positive exponent is more prevailing and therefore we will use this notation.

As a simple example consider the z-transforms of the sequences $[t_0, t_1, t_2, t_3]$ and $[u_0, u_1, u_2, u_3]$ computed as

$$T(z) = t_0 + t_1 z + t_2 z^2 + t_3 z^3, \quad U(z) = u_0 + u_1 z + u_2 z^2 + u_3 z^3. \tag{2}$$

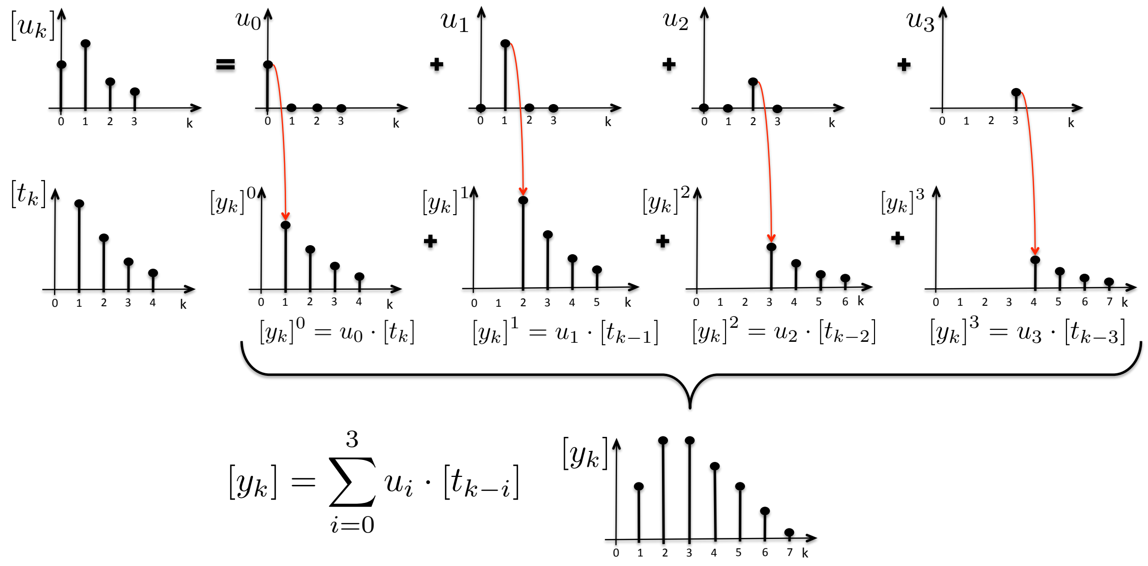
$[u_k]$
u_0
u_1
u_2
u_3
$[t_k]$
$[y_k]^0$
$[y_k]^1$
$[y_k]^2$
$[y_k]^3$
$[y_k]^0 = u_0 \cdot [t_k]$
$[y_k]^1 = u_1 \cdot [t_{k-1}]$
$[y_k]^2 = u_2 \cdot [t_{k-2}]$
$[y_k]^3 = u_3 \cdot [t_{k-3}]$
$[y_k] = \sum_{i=0}^{3} u_i \cdot [t_{k-i}]$
$[y_k]$
k

Figure 2: Computing the output signal of an LTI system with convolutions. The first row denotes the decomposition of an input sequence $[u_k]$ into the sum of individual impulses u_i, which are shifted in time. Each of the individual impulses generates a shifted version of the impulse response, which weighted with the value of the corresponding impulse u_i. These impulse responses are shown in the second row. Finally, the output signal $[y_k]$ is generated as the sum of the individually weighted and shifted impulse responses.

We denote the the z-transform of the input sequence by $U(z)$ and the z-transform of the impulse response of the system by $T(z)$. The z-transform of the output signal $Y(z)$ is computed by multiplying the corresponding z-transforms

$$Y(z) = T(z) \cdot U(z). \tag{3}$$

For the current example this amounts to computing the values y_k in the z-transform

$$Y(z) = y_0 + y_1 z + y_2 z^2 + y_3 z^3 + y_4 z^4 + y_5 z^5 + y_6 z^6, \tag{4}$$

from which we can read off the output sequence $[y_0, y_1, y_2, y_3, y_4, y_5, y_6]$. The values of the z-transform of $Y(z)$ are computed as the convolution of the coefficient vectors for $U(z)$ and $T(z)$.

$$Y(z) = (t_0 u_0) + (t_1 u_0 + t_0 u_1) z + (t_2 u_0 + t_1 u_1 + t_0 u_2) z^2 + (t_3 u_0 + t_2 u_1 + t_1 u_2 + t_0 u_3) z^3 + \dots$$

$$\dots + (t_3 u_1 + t_2 u_2 + t_1 u_3) z^4 + (t_3 u_2 + t_2 u_3) z^5 + (t_3 u_3) z^6$$

After all, the z-transform is a nice tool for calculating convolutions by hand if it is restricted to relatively short sequences and short filters, as otherwise the manual work will be overwhelming. We still need an efficient method for computing the convolution of two finite sequences.

4 Linear Convolution

The output sequence $[y_k]$ is determined by the convolution operation. We now discuss the computational task to compute the convolution of two signals in an efficient way. Let's consider a finite, discrete-time sequence $[u_k], k = 0, 1, 2, \dots m-1$ of length m. We feed this sequence as the input to a (discrete-time) linear, time-invariant system that is described by its associated impulse response $[t_k], k = 1, 2, \dots n-1$ of length n. For now we restrict the discussion to finite time series to keep things simple. We can compute the output sequence $[y_k]$ of the linear system as the *linear* convolution of the two sequences $[t_k]$ and $[u_k]$, denoted by

$$[y_k] = [t_k] \star [u_k] = \sum_{i=0}^{m-1} [t_{k-i}] \cdot u_i, \quad k = 0, 1, \dots m+n-2.$$

with an input sequence $[u_k]$ of length m and an impulse response $[t_k]$ of length n, then the length of the output signal is $N = m + n - 1$.

As an example, we manually convolve an input signal $[u_k]$ of length $m = 4$ with an impulse response $[t_k]$

of length $n = 4$ to produce an output signal $[y_k]$ of length $N = m + n - 1 = 7$,

	0	0	0	u_0	u_1	u_2	u_3	0	0	0
	t_3	t_2	t_1	t_0						
y_0				u_0t_0						
		t_3	t_2	t_1	t_0					
y_1				u_0t_1	$+u_1t_0$					
			t_3	t_2	t_1	t_0				
y_2				u_0t_2	$+u_1t_1$	$+u_0t_2$				
				t_3	t_2	t_1	t_0			
y_3				u_0t_3	$+u_1t_2$	$+u_2t_1$	$+u_3t_0$			
					t_3	t_2	t_1	t_0		
y_4					u_1t_3	$+u_2t_2$	$+u_3t_1$			
						t_3	t_2	t_1	t_0	
y_5						u_2t_3	$+u_3t_2$			
							t_3	t_2	t_1	t_0
y_6							u_3t_3			

4.1 Linear Convolution as Matrix-Vector Operation

We can use the elements of the input sequence $[u_k]$ to form the m-dimensional vector, and the entries of the output sequence $[y_k]$ can also be summarized in a corresponding output vector

$$u = \begin{bmatrix} u_0 \\ u_1 \\ \\ u_{m-1} \end{bmatrix}, \quad u \in \mathbb{R}^m, \qquad t = \begin{bmatrix} t_0 \\ t_1 \\ \vdots \\ t_{n-1} \end{bmatrix}, \quad t \in \mathbb{R}^n, \qquad y = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{m+n-2} \end{bmatrix}, \quad y \in \mathbb{R}^{m+n-1}.$$

We convert the impuls response vector t into a matrix T of dimension $(m + n - 1) \times n$. For an example with $n = 4$ this looks like

$$t = \begin{bmatrix} t_0 \\ t_1 \\ \vdots \\ t_3 \end{bmatrix} \mapsto T = \begin{bmatrix} t_0 & & & \\ t_1 & t_0 & & \\ t_2 & t_1 & t_0 & \\ t_3 & t_2 & t_1 & t_0 \\ & t_3 & t_2 & t_1 \\ & & t_3 & t_2 \\ & & & t_3 \end{bmatrix}.$$

Using these vectors and the matrix we can replace the tedious convolution sum operation, with a matrix-vector multiplication $y = T \cdot u$, e.g. for $m = 4$ and $n = 4$ we have

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} = \underbrace{\begin{bmatrix} t_0 & & & \\ t_1 & t_0 & & \\ t_2 & t_1 & t_0 & \\ t_3 & t_2 & t_1 & t_0 \\ & t_3 & t_2 & t_1 \\ & & t_3 & t_2 \\ & & & t_3 \end{bmatrix}}_{T} \cdot \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix}. \tag{5}$$

Note that shifted versions of the impulse-response constitute the columns of the matrix T, which results in having identical entries along diagonals. The matrix T with this specific structure is a finite dimensional representation of a more general *convolution operator* and is called a *Toeplitz* matrix after the German mathematician Otto Toeplitz.

The current Toeplitz matrix is a rectangular matrix, which is not invertible. We also cannot determine its eigenvalues and eigenvectors. Furthermore, in order to enable efficient algorithms for computing convolutions we complete the matrix appropriately. We explain the details in the next section. The computation of the convolution is represented as a matrix-vector product, which requires $(m+n-1)\cdot m = m^2+mn-m$ operations, which amounts for an asymptotic complexity of $\mathcal{O}(m^2)$ operations. We are interested in reducing the complexity as the Toeplitz structure of T provides opportunities for optimization.

4.2 Cyclic Convolution and Cyclic Toeplitz Matrix

Let's have a look at a particular class of matrices, which are closely related to the Toeplitz matrix of the previous section. This class is referred to as *Circulant Matrices* or *Cyclic Toeplitz Matrices.* Using the entries of the impulse response we can build a representative T_c of this family, e.g.

$$T_c = \begin{bmatrix} t_0 & t_3 & t_2 & t_1 \\ t_1 & t_0 & t_3 & t_2 \\ t_2 & t_1 & t_0 & t_3 \\ t_3 & t_2 & t_1 & t_0 \end{bmatrix}.$$

The reader will immediately recognize the cyclic property of this matrix. With this cyclic Toeplitz matrix we can compute the *Cyclic Convolution* of the signals $[t_k]$ and $[u_k]$ as

$$[y_k] = [t_k] \otimes [u_k], \quad y = T_c u.$$

The elements of the output sequence come out as

$$y = \begin{bmatrix} t_0u_0 + t_3u_1 + t_2u_2 + t_3u_1 \\ t_1u_0 + t_0u_1 + t_1u_2 + t_2u_1 \\ t_2u_0 + t_1u_1 + t_0u_2 + t_3u_1 \\ t_3u_0 + t_2u_1 + t_1u_2 + t_0u_1 \end{bmatrix},$$

which clearly differs substantially from the result of the linear convolution. Using a cyclic Toeplitz matrix, we still would like to compute the linear convolution of $[t_k]$ and $[u_k]$. To achieve this, we build a cyclic Toeplitz matrix with an impulse response vectors $[t_k]$, which we pad by the necessary number of zeroes such that T_c becomes a $(m+n-1)\times(m+n-1)$-matrix. We need to also pad the input vector u appropriately with additional zeros, such that the additional columns in T_c do not modify the output vector y. We still denote the zero-padded input vector as u. With this recipe we can calculate the linear convolution correctly using actually a cyclic convolution $y = T_c \cdot u$ as

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} = \left[\begin{array}{cccc|ccc} t_0 & & & & t_3 & t_2 & t_1 \\ t_1 & t_0 & & & & t_3 & t_2 \\ t_2 & t_1 & t_0 & & & & t_3 \\ t_3 & t_2 & t_1 & t_0 & & & \\ & t_3 & t_2 & t_1 & t_0 & & \\ & & t_3 & t_2 & t_1 & t_0 & \\ & & & t_3 & t_2 & t_1 & t_0 \end{array}\right] \cdot \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{6}$$

The cyclic extension of the Toeplitz matrix is in close relation with the known effects of periodic replication of the signal and its Fourier spectrum when processing signals, which have been sampled in the time domain and the frequency domain.

So why do we jump through all these hoops to make a simple operation like a linear convolution seemingly more complicated by turning it into a cyclic convolution involving zero-padded vectors and matrices? I will explain this in the next section. Bear in mind, that after all we are seeking efficiency in computation and hence we are looking for ways to reduce computations.

5 Fourier Transformation and Frequency Domain

5.1 Similarity Transformation

Since we have now established the cyclic Toeplitz matrix and its use for computing the linear convolution we want to exploit the property that T_c is a square matrix. This allows us to do further analysis of the properties of this family of matrices.

If we multiply both sides of the equation $y = T_c u$ from the left with a non-singular matrix Q, we arrive at

$$Q \cdot y = Q \cdot T_c \cdot u. \tag{7}$$

As a next step we insert the identity matrix $\mathbf{1}_n = Q^{-1}Q$ between the factors T_c and u on the right hand side of Equation (7). This leads us to

$$Q \cdot y = Q_c \cdot Q^{-1} \cdot Q \cdot u. \tag{8}$$

Inserting a few brackets for improved readability we get

$$(Q \cdot y) = (Q \cdot T_c \cdot Q^{-1}) \cdot (Q \cdot u), \tag{9}$$

where we can read off the following abbreviated notation

$$Y = \Lambda \cdot U, \tag{10}$$

where the quantities Y, Λ and U are defined as

$$Y := Q \cdot y, \qquad U := Q \cdot u, \qquad \Lambda := Q \cdot T_c \cdot Q^{-1}. \tag{11}$$

The identity

$$\Lambda := Q \cdot T_c \cdot Q^{-1} \tag{12}$$

can easily be identified as a similarity transformation of T_c by Q. The similarity transformation implies that the matrices T_c and Λ share the same eigenvalues. If the matrix Q where the eigenvectors of T_c, the Λ would turn out to be a diagonal matrix with the eigenvalues of T_c on its main diagonal (of course only of T_c is diagonalizable). So, we are interested to determine the eigenvalues and eigenvectors of the Cyclic Toeplitz matrix.

5.2 Eigenvalue Decomposition of Cyclic Toeplitz Matrices

The specific structure of the cyclic Toeplitz matrix is also called a *Circulant*. We want to determine a eigenvalue decomposition for a cyclic Toeplitz matrix, i.e. we want to compute eigenvectors x and eigenvalues λ (solutions to $T_c x = \lambda x$) for a cyclic Toeplitz matrix, given as

$$T_c = \begin{bmatrix} t_0 & t_{N-1} & t_{N-2} & \dots & t_1 \\ t_1 & t_0 & t_{N-1} & \dots & t_2 \\ t_2 & t_1 & t_0 & \dots & t_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{N-1} & t_{N-2} & t_{N-3} & \dots & t_0 \end{bmatrix}. \tag{13}$$

Let q denote a root of the scalar equation $q^N = 1$, where we for now can take $N = m + n - 1$ and set

$$x = \begin{bmatrix} q^0 & q^1 & q^2 & \dots & q^{N-1} \end{bmatrix}'.$$

We then compute

$$z = T_c \cdot x = \begin{bmatrix} z_0 & z_1 & z_2 & \dots & z_{N-1} \end{bmatrix}'.$$

Looking at the first entry of the resulting vector z determined as

$$z_0 = t_0 + t_{N-1}q^1 + t_{N-2}q^2 + \dots + t_1 q^{N-1}$$

we observe that z_0 satisfies the following system of equations

$$\begin{aligned} z_0 &= t_0 + t_{N-1}q^1 + t_{N-2}q^2 + \dots + t_1 q^{N-1} \\ z_1 = z_0 q^1 &= t_1 + t_0 q^1 + t_{N-1}q^2 + \dots + t_2 q^{N-1} \\ \vdots \quad & \quad \vdots \\ z_{N-1} = z_0 q^{N-1} &= t_{N-1} + t_{N-2}q^1 + t_{N-3}q^2 + \dots + t_0 q^{N-1}, \end{aligned}$$

which we can summarize compactly as

$$z_0 \cdot x = T_c \cdot x.$$

It follows from this that $\lambda = z_0$ is a characteristic root (eigenvalue) of T_c with the associated characteristic vector (eigenvector) x. Since the equation $q^N = 1$ has N distinct roots $\lambda_i, i = 0, 2, \dots N-1$, we see that we obtain N distinct characteristic vectors $x^i, i = 0, 1, 2, \dots N-1$. The value $q = e^{j2\pi/N}$ is a solution to the equation $q^N = 1$. Consequently, we have the complete set of characteristic roots and vectors in this way, i.e.

$$T_c \cdot x^i = \lambda_i \cdot x^i$$

holds. The set of all eigenvectors x_i can be put together as the columns of a matrix

$$Q = \left[x^0, x^1, x^2, \dots, x^{N-1}\right] = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & q & q^2 & \dots & q^{N-1} \\ 1 & q^2 & q^4 & \dots & q^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & q^{N-1} & & \dots & q^{(N-1)^2} \end{bmatrix}.$$

With this special choice for the matrix Q the Equation (12) represents the *Eigenvalue Decomposition* of T_c with

$$\Lambda = Q^{-1} \cdot T_c \cdot Q = \begin{bmatrix} \lambda_0 & & & 0 \\ & \lambda_1 & & \\ & & \ddots & \\ 0 & & & \lambda_{N-1} \end{bmatrix}. \tag{14}$$

That is, Λ contains the eigenvalues of T_c as its diagonal entries and the matrix Q contains the corresponding eigenvectors. That implies that for computing the Eigenvalue decomposition of the cyclic Toeplitz matrix T_c we already have the corresponding eigenvectors given a priori as the columns of the matrix Q. With those quantities given, computing the pertaining eigenvalues is an easy task.

5.3 Discrete Fourier Transformation

Let us have a closer look at the matrix Q, which is composed of the characteristic vectors $x^i, i = 1, 2, \ldots N-1$,

$$Q = \frac{1}{\sqrt{N}} \cdot \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & q & q^2 & \ldots & q^{N-1} \\ 1 & q^2 & q^4 & \ldots & q^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & q^{N-1} & & \ldots & q^{(N-1)^2} \end{bmatrix}, \tag{15}$$

which we have now conveniently normalized by a factor $1/\sqrt{N}$ to arrive at a matrix Q having a number of special properties, such as

- Unitarity: $Q^H Q = \mathbf{1}_N \quad \Rightarrow \quad Q^H = Q^{-1}$
- Permutation: $Q^2 = P = P'$
- Cyclic: $Q^3 = Q^H, \quad Q^4 = \mathbf{1}_N$.

This matrix Q is identified as the matrix of the *D*iscrete Fourier Transform (DFT) for sequences of length N. That means that the eigenvalue decomposition of the cyclic Toeplitz matrix T_c corresponds with the Discrete Fourier Transform. With view to equation 12 we can identify that the eigenvalues of T_c collected in the matrix Λ represent actually the Fourier spectrum of the (zero-padded) sequence $[t_k]$. (Note that the zero-padding is not an essential part of this process). We can identify the matrix Q to represent the *Discrete Fourier Transform*, such that the Fourier spectrum of the sequences $[u_k]$ and $[t_k]$ are determined by

$$diag\,\{\Lambda\} = Q \cdot t \leftrightarrow \mathcal{DFT}\{[t_k]\}, \quad U = Q \cdot u \leftrightarrow \mathcal{DFT}\{[u_k]\}, \quad Y = Q \cdot y \leftrightarrow \mathcal{DFT}\{[y_k]\}. \tag{16}$$

The convolution theorem associated with the Fourier Transformation appears in the form

$$[y_k] = \mathcal{DFT}^{-1}\{\mathcal{DFT}\{[t_k]\} \cdot \mathcal{DFT}\{[u_k]\}\} = [t_k] \star [u_k] \tag{17}$$

then becomes $Y = \Lambda \cdot U$ or more explicitely

$$\begin{bmatrix} Y_0 \\ Y_1 \\ \vdots \\ Y_{N-1} \end{bmatrix} = \begin{bmatrix} \lambda_0 & & & 0 \\ & \lambda_1 & & \\ & & \ddots & \\ 0 & & & \lambda_{N-1} \end{bmatrix} \cdot \begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ U_{N-1} \end{bmatrix}. \tag{18}$$

This just says that the convolution of sequences in the time domain corresponds to the multiplication of the corresponding spectra in the frequency (Fourier) domain. As engineers we have learned to appreciate this powerful statement. However, when looking at the complexity of computing the convolution of sequences using Fourier techniques, we can identify that computing the Fourier spectra for the sequences $[u_k]$ and $[t_k]$ comes as a matrix vector multiplication and hence requires $\mathcal{O}(N^2)$ operations ($N = m + n - 1$). The multiplication of the Fourier spectra takes $\mathcal{O}(N)$ operations and the back transformation of the resulting spectrum into the time domain is again an $\mathcal{O}(N^2)$ operation. So, after all these considerations and derivations we have now established the frequency domain, but we it still takes too much operations to calculate a linear convolution.

But there is a way to continue. The previously derived identities are the basis on which the technique of *Fast Convolution* is based on. For real and efficient computations of the DFT we use an fast algorithm, called the *Fast Fourier Transform* or *FFT*. The FFT computes the eigenvalues of the cyclic Toeplitz matrix T_c using $\mathcal{O}(N \log N)$ arithmetic operations, which is much more efficient than a straight matrix-vector multiplication. The FFT exploits special properties of the matrix Q in combination with a clever devide-and-conquer approach to matrix multiplication. But that thing, we will discuss in the next section.

6 The Fast Fourier Transform

The Fast Fourier Transform (FFT) is a collection of computationally efficient methods to compute the DFT of order N, assuming that N is a product of smaller integers $N = r \cdot s$. If r and s are themselves products of even smaller numbers then this process can be further exploited recursively. Often N will be a power of 2, e.g. $N = 2^n$ or even $N = 4^n$ and the resulting scheme will become especially efficient, but the case where r and s are primes is in itself also interesting. It is this numerical efficiency that has made the FFT a method of choice for a number of tasty signal processing problems such as convolution, non parametric spectral estimation, autocorrelation and the solution of certain types of structured matrix equations.

The original discovery of the FFT is due to Tuckey and Cooley [3] in 1965, but meanwhile many contributions have been made detailing algorithmic improvements and use in various circumstances, various books have been written about it and a lot of experience has been obtained since it has become the obvious method of choice to execute Fourier Transforms concretely.

There is a particularly attractive way to introduce the FFT just by looking at symmetries in the DFT matrix. This is the way we shall follow. We take the smallest possible non trivial example: $N = 12 = 3 \cdot 4$, and show how FFT-12 is reduced to a number of FFT-3's and FFT-4's. The method we follow will be perfectly general and the reader will immediately understand how the method can be generalized to the $N = r * s$ case.

6.1 Matrix of Powers in q^{-1}

Embarking on the $N = 12 = 3 \cdot 4$ case we have the DFT-Matrix (see also [?])

$$Q_{12} = \frac{1}{\sqrt{12}} \cdot \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & q & q^2 & \dots & q^{11} \\ 1 & q^2 & q^4 & \dots & q^{22} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & q^{11} & & \dots & q^{121} \end{bmatrix},$$

Exploiting the cyclic property of the powers of $q^{12} = 1$ the matrix entries can be simplified to

$$Q_{12} = \frac{1}{\sqrt{12}} \cdot \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & q & q^2 & \dots & q^{11} \\ 1 & q^2 & q^4 & \dots & q^{10} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & q^{11} & q^{10} & \dots & q^1 \end{bmatrix}.$$

Instead of writing out the Q_{12} completely in matrix form, we write all the powers that appear in the Q_{12} matrix, also in a 12×12 matrix form. This matrix shows the symmetries that we shall exploit in the algorithm nicely

$$Q_{12} = \frac{1}{\sqrt{12}} q\hat{.} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 0 & 2 & 4 & 6 & 8 & 10 & 0 & 2 & 4 & 6 & 8 & 10 \\ 0 & 3 & 6 & 9 & 0 & 3 & 6 & 9 & 0 & 3 & 6 & 9 \\ 0 & 4 & 8 & 0 & 4 & 8 & 0 & 4 & 8 & 0 & 4 & 8 \\ 0 & 5 & 10 & 3 & 8 & 1 & 6 & 11 & 4 & 9 & 2 & 7 \\ 0 & 6 & 0 & 6 & 0 & 6 & 0 & 6 & 0 & 6 & 0 & 6 \\ 0 & 7 & 1 & 8 & 2 & 9 & 3 & 10 & 4 & 11 & 5 & 0 \\ 0 & 8 & 4 & 0 & 8 & 4 & 0 & 8 & 4 & 0 & 8 & 4 \\ 0 & 9 & 6 & 3 & 0 & 9 & 6 & 3 & 0 & 9 & 6 & 3 \\ 0 & 10 & 8 & 6 & 4 & 2 & 0 & 10 & 8 & 6 & 4 & 2 \\ 0 & 11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \end{bmatrix} \tag{19}$$

In this write up of Q_{12} we have used the same notation as in the previous chapter, namely $q = 2^{2\pi j/12}$, and a MATLAB-like notation for point wise exponentiation ($\hat{.}$) of matrix elements. We use the fact that the exponents of q may be restricted to the range $0 \dots 11$ by applying a 'mod 12' calculation. Various symmetries jump into view in the matrix of exponents. The issue is how to exploit these symmetries for efficient computations.

Our goal is the construction of a computational schema for the DFT, i.e. the computation of the Discrete Fourier Spectrum of the input sequence $[u_k]$, which is captured in the the vector u. This computation amounts to perform the matrix-vector product

$$U = Q_{12} \cdot u,$$

in which U and u are vectors of dimension $N = 12$. The main effect of the mentioned symmetries is that in this matrix-vector product the same sums appear many times. A color schema of repetitions in the matrix of exponents is shown in Figure 3.

For example, $u_0+u_4+u_8$ appears in rows 0, 3, 6 and 9 (zero exponent), just as well as the sum $u_1+u_5+u_9$, except that in the latter case the sum is additionally multiplied with a changing factor in rows 3, 6 and 9. The best way to deal with this is to reorder the matrix so as to bring potential symmetries together.

6.2 Column permutations bringing sums together

We perform a column permutation on Q_{12}. The permutation is simply: 0, 4, 8, 1, 5, 9, 2, 6, 10, 3, 7, 11 (i.e. row 0 remains, row 4 becomes row 1 etc.). Writing Π_4 for the corresponding permutation matrix we

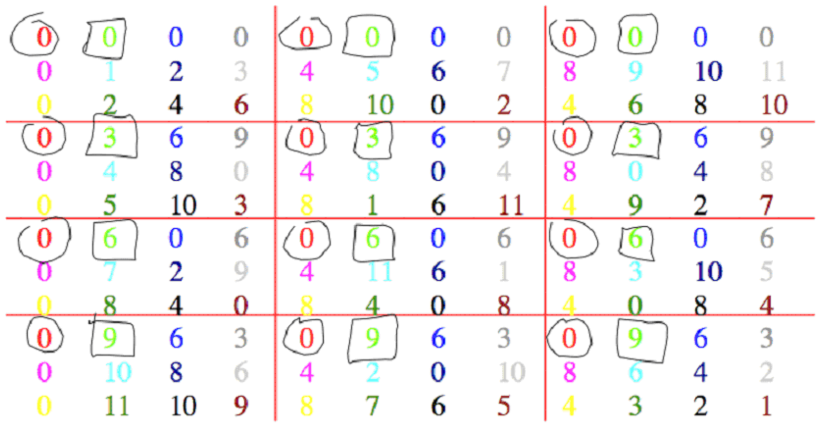

Figure 3: Schematic Procedure for FFT.

obtain for the resulting schema

$$Q_{12}\Pi_4 = q\hat{:}\left[\begin{array}{ccc|ccc|ccc|ccc}
0&0&0&0&0&0&0&0&0&0&0&0\\
0&4&8&1&5&9&2&6&10&3&7&11\\
0&8&4&2&10&6&4&0&8&6&2&10\\
\hline
0&0&0&3&3&3&6&6&6&9&9&9\\
0&4&8&4&8&0&8&0&4&0&4&8\\
0&8&4&5&1&9&10&6&2&3&11&7\\
\hline
0&0&0&6&6&6&0&0&0&6&6&6\\
0&4&8&7&11&3&2&6&10&9&1&5\\
0&8&4&8&4&0&4&0&8&0&8&0\\
\hline
0&0&0&9&9&9&6&6&6&3&3&3\\
0&4&8&10&2&6&8&0&4&6&10&2\\
0&8&4&11&7&3&10&6&2&9&5&1
\end{array}\right].$$

A subdivision in $3Times3$ blocks shows the pre-eminence in the first three columns of the block

$$Q_3 = q\hat{:}\begin{bmatrix}0&0&0\\0&4&8\\0&8&4\end{bmatrix} = q^4\hat{:}\begin{bmatrix}0&0&0\\0&1&2\\0&2&1\end{bmatrix},$$

which we identify easily as the FFT 3 block, since $e^{2\pi 4/12} = e^{2\pi/3}$. Look at the second block of three columns, e.g.

$$\begin{bmatrix}3&3&3\\4&8&0\\5&1&9\end{bmatrix} = \begin{bmatrix}3&&\\&4&\\&&5\end{bmatrix}\star\begin{bmatrix}0&0&0\\0&4&8\\0&4&8\end{bmatrix},$$

in which the matrix multiplication $|star$ has a special interpretation due to the exponential representation used. The reader will figure out the mystery

Let us now post multiply the matrix obtained so far with the inverse of a block-diagonal matrix consisting of four Q_3 blocks, and let us call the block matrix $D_3^{[4]}$

$$D_3^{[4]} = \begin{bmatrix}Q_3&&&\\&Q_3&&\\&&Q_3&\\&&&Q_3\end{bmatrix}.$$

Taking the observation made so far into account we now obtain

$$Q_{12}\Pi_4(D_3^{[4]})^{-1} = q\hat{:}\left[\begin{array}{ccc|ccc|ccc|ccc}
0&&&0&&&0&&&0&&\\
&0&&&1&&&2&&&3&\\
&&0&&&2&&&4&&&6\\
\hline
0&&&3&&&6&&&9&&\\
&0&&&4&&&8&&&0&\\
&&0&&&5&&&10&&&3\\
\hline
0&&&6&&&0&&&6&&\\
&0&&&7&&&2&&&9&\\
&&0&&&8&&&4&&&0\\
\hline
0&&&9&&&6&&&6&&\\
&0&&&10&&&8&&&6&\\
&&0&&&11&&&10&&&9
\end{array}\right]$$

in which the empty entries are zero. Again, an observation as at the start of our operations imposes itself on us, and we reorder the columns, this time by the column permutation 0,3,6,9,1,4,7,10,2,5,8,11, which we call Π_3

$$Q_{12}\Pi_4(D_3^{[4]})^{-1}\Pi_3 = q\hat{} \left[\begin{array}{cccc|cccc|cccc}
0 & 0 & 0 & 0 & & & & & & & & \\
 & & & & 0 & 1 & 2 & 3 & & & & \\
 & & & & & & & & 0 & 2 & 4 & 6 \\
\hline
0 & 3 & 6 & 9 & & & & & & & & \\
 & & & & 0 & 4 & 8 & 0 & & & & \\
 & & & & & & & & 0 & 5 & 10 & 3 \\
\hline
0 & 6 & 0 & 6 & & & & & & & & \\
 & & & & 0 & 7 & 2 & 9 & & & & \\
 & & & & & & & & 0 & 8 & 4 & 0 \\
\hline
0 & 9 & 6 & 3 & & & & & & & & \\
 & & & & 0 & 10 & 8 & 6 & & & & \\
 & & & & & & & & 0 & 11 & 10 & 9
\end{array}\right]$$

Again we use the convention that empty entries are actually zero. This time a new phenomenon appears, similar operations have to be done as before, but now regrouping on the rows and block multiplication on the left. Let us first observe that column five equals column 1 multiplied with q, column six is column two multiplied with q^2 etc. These factors are the so called *twiddle factors*. Let us define the block diagonal matrix

$$W = q\hat{} \left[\begin{array}{cccccccccccc}
1 & & & & & & & & & & & \\
 & 1 & & & & & & & & & & \\
 & & 1 & & & & & & & & & \\
 & & & 1 & & & & & & & & \\
 & & & & 1 & & & & & & & \\
 & & & & & q & & & & & & \\
 & & & & & & q^2 & & & & & \\
 & & & & & & & q^3 & & & & \\
 & & & & & & & & 1 & & & \\
 & & & & & & & & & q^2 & & \\
 & & & & & & & & & & q^4 & \\
 & & & & & & & & & & & q^6
\end{array}\right]$$

and let us regroup the rows by permutation. What is the permutation matrix that we should apply now? Let us first determine Π_4, which permutes the columns in the order 0,4,8,1 etc. Applying the rules of matrix multiplication we obtain easily

$$\Pi_4 = \left[\begin{array}{ccc|ccc|ccc|ccc}
1 & & & & & & & & & & & \\
 & & & 1 & & & & & & & & \\
 & & & & & & 1 & & & & & \\
 & & & & & & & & & 1 & & \\
\hline
 & 1 & & & & & & & & & & \\
 & & & & 1 & & & & & & & \\
 & & & & & & & 1 & & & & \\
 & & & & & & & & & & 1 & \\
\hline
 & & 1 & & & & & & & & & \\
 & & & & & 1 & & & & & & \\
 & & & & & & & & 1 & & & \\
 & & & & & & & & & & & 1
\end{array}\right]$$

Looking carefully at this matrix we see that when applied to the left of a matrix it will exchange the rows of the latter in the order 0,3,6,9,1 etc. Hence this is the matrix we are looking for to regroup the rows. A little more exploration shows that in fact $\Pi_4\Pi_3 = \Pi_3\Pi_4$, hence $\Pi_3 = \Pi_4^{-1}$ - a fact that we shall use soon. Applying now W to the right of the result so far and Π_4 to the left we find

$$\Pi_4 Q_{12}\Pi_4(D_3^{[4]})^{-1}\Pi_3 W = q\hat{:} \left[\begin{array}{cccc|cccc|cccc}
0&0&0&0&&&&&&&& \\
0&3&6&9&&&&&&&& \\
0&6&0&6&&&&&&&& \\
0&9&6&3&&&&&&&& \\ \hline
&&&&0&0&0&0&&&& \\
&&&&0&3&6&9&&&& \\
&&&&0&6&0&6&&&& \\
&&&&0&9&6&3&&&& \\ \hline
&&&&&&&&0&0&0&0 \\
&&&&&&&&0&3&6&9 \\
&&&&&&&&0&6&0&6 \\
&&&&&&&&0&9&6&3
\end{array}\right]$$

a result that we recognize as $D_3^{[4]}$ - three diagonal blocks of Q_4. Hence, using $\Pi_4^{-1} = \Pi_3$ as well as vice versa and inverting the necessary matrices we find

$$Q_{12} = \Pi_3 D_4^{[3]} W^{-1}\Pi_4 D_3^{[4]}\Pi_3.$$

6.3 The resulting computational schema

In the previous section Q_{12} has been reduced to three permutations, four Q_3's, three Q_4's and a number of twiddle factors (six if we do not count multiplications by 1, but of those six, two more are also trivial because they reduce to multiplications by either -1 or j). How does this translate to a signal flow graph? To see that, we just imagine that we multiply Q_{12} with an arbitrary signal vector u to the right. The first operation on u just permutes the entries, next four Q_3 blocks come into play etc. This produces the schema depicted in Figure 4.

6.4 Computational Complexity

The complexity count takes into account that multiplications with a 1, -1, j or -j do not count (nor do sign changes count because they can be incorporated in the summations), and that

$$Q_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

hence results in no multiplications, but 12 additions (for each row 3). Each Q_3 has 4 complex multiplications and 6 additions, while multiplication with the twiddle factors result in four complex multiplications. The overall count is then for the FFT 20 multiplications and 60 additions. Already in this instance the count is much in favor of the FFT as compared to the DFT (whose count we leave to the reader).

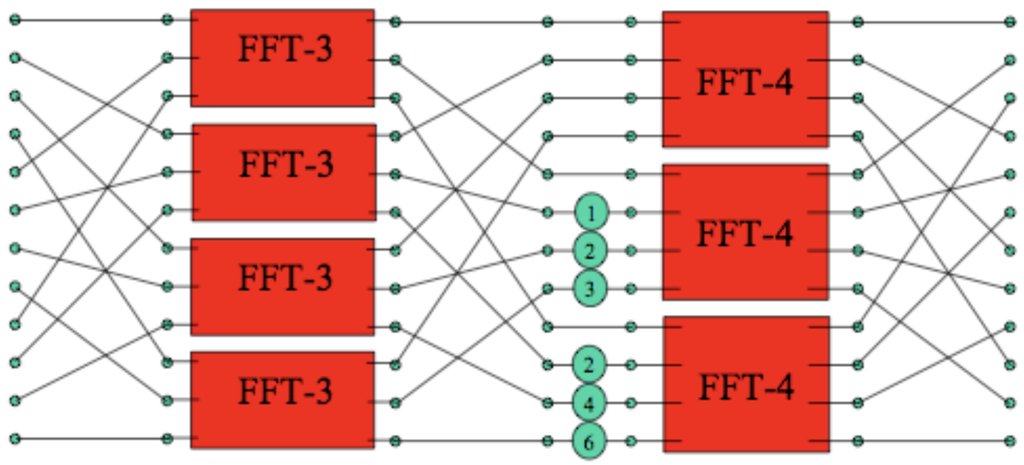

Figure 4: Signal Flow Graph for Computing the FFT. Twiddle Factors colored are colored in annotated with their respective coefficient of q.

6.4.1 Generalization

The theory so far generalizes easily to the product $N = r \cdot s$ and two (dual) schemas will be obtained, one for $r \cdot s$ and one for $s \cdot r$. Using the notation developed in this chapter and adapted to the new situation we can write

$$Q_{r\cdot s} = \Pi_r D_s^{[r]} W^{-1} \Pi_s D_r^{[s]} \Pi_r,$$

where some attention has to be devoted to the structure of the twiddle factor matrix. It will consist of r diagonal blocks of dimension s

$$W = diag\,\{W_0, W_1, \ldots, W_{r-1}\}$$

with

$$W_j = diag\,\{1, q^j, \ldots, q^{s-1}\},$$

in particular W_0 is a unit matrix and the maximum power of q is $(r-1)(s-1)$.

7 Computing in the Fourier Domain

The convolution operation costs $\mathcal{O}(N^2)$ operations, if N is the length of the signals we are working on. Alternatively, exploiting the convolution theorem of the Discrete Fourier Transformation we can compute the output signal of a linear time-invariant system in the frequency domain. The Discrete Fourier Transformation of a given signal can be computed very efficient way by the *Fast Fourier Transform* (FFT). This leads to the so-called *Fast Convolution* method, which requires $\mathcal{O}(N \log N)$ operations.

This denotes a significant saving in computations when comparing the Fast Convolution with the direct execution of the convolution sum. Figure 5 shows the detour through frequency domain for computing the convolution.

This frequency domain method using the FFT works very well, is well established and it's efficiency can be regarded as one of the major cornerstones for the tremendous success of digital signal processing during the past 30 years. However, it is based on the assumption that the systems involved are linear and time-invariant. If one of these two assumptions is violated, then the use of frequency domain tools is no longer possible.

If the system is time-varying, then the impulse response changes with time. That implies that the columns in the convolution operator T are not shifter versions of the impulse response $[t_k]$ and hence the operator looses its Toeplitz property. Also, the cyclic Toeplitz matrix does not exhibit the particular structure anymore, which is the cornerstone for the matrix Q to consist of the eigenvectors of T_c and hence the convolution theorem of the Discrete Fourier Transform does not hold anymore.

Successfully and efficiently computing in the Fourier domain is also fitting only if the impulse responses $[t_k]$ describing the input-output behavior of the system has finite length. In signal processing applications engineers use recursive filters, which have a finite number of parameters, but which produce an impulse response of infinite length.

Literatur

[1] G. Strang. *Computational Science and Engineering.* Wellesley-Cambridge Press, 2007.

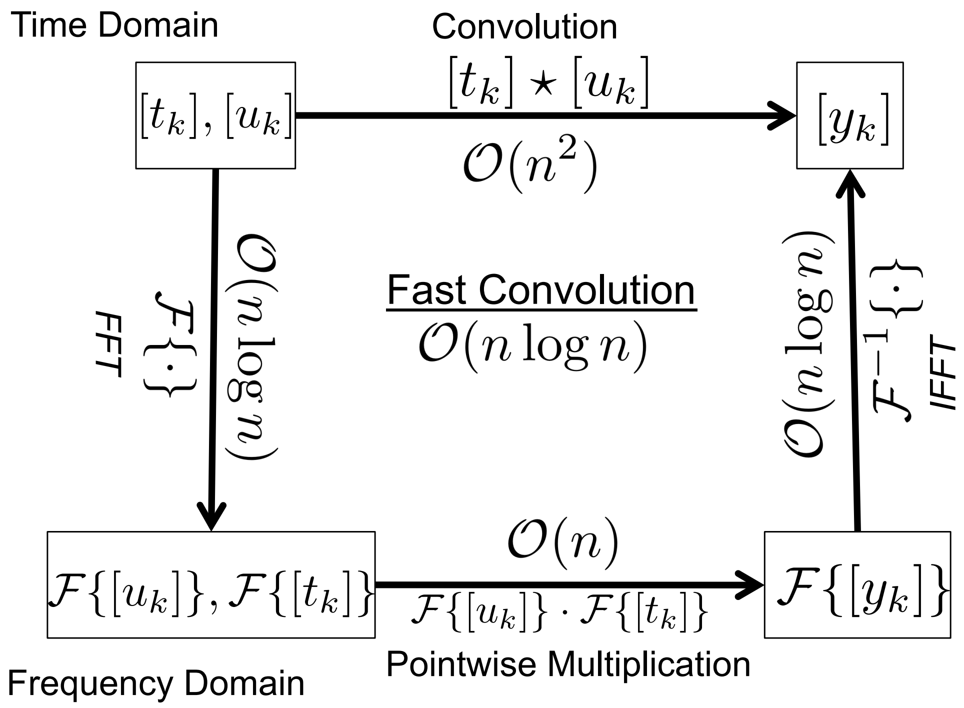

Figure 5: Schematic Procedure for Computing Fast Convolutions.

[2] P. Dewilde, A.-J. van der Veen. *Time-Varying Systems and Computations.* Kluwer Academic Publishers, 1998.

[3] James W. Cooley, John W. Tukey. *An algorithm for the machine calculation of complex Fourier series*. Math. Comput. 19, 1965, S. 297-301.