Multiorder multidimensional systems: Computation of the transfer function using the DFT

A Thesis
Presented to
The Academic Faculty

by

Marinos Theodorou Michael

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Computer Science

Department of Computer Science
Montclair State University
May 2006
Multiorder multidimensional systems: Computation of the transfer function using the DFT

by

Marinos Theodorou Michael

A Master's Thesis Submitted to the Faculty of
Montclair State University
In Partial Fulfillment of the Requirements
For the Degree of
Master of Science
May 2006
To my family and friends,

for their continued support
ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis advisor, Prof. G E. Antoniou for giving me the opportunity to work on this exciting project and for his faith, dedication, encouragement, guidance and support throughout the course of my master research. I have been working with Prof. G. E. Antoniou since the first day I came to the U.S. I benefited from his guidance in every aspect during my Master’s study. It was his wide knowledge of Computer Engineering especially on Multidimensional Systems Analysis, creative thoughts and sparkling of fresh ideas, dedication to scientific excellence, and most important, his pure love and enthusiasm toward scientific research, that had led to every progress in my Master research. I deeply appreciate all his invaluable help, the care he showed for me, and the freedom he gave me.

I would also like to thank Dr. Angel Gutierrez and Dr. Stefan Robila who gladly agreed to serve as my thesis committee and taking time out of their busy schedules to evaluate my work.

I would also like to express my appreciation to all my friends and colleagues, here at Montclair State University, for their help in successful completion of this thesis.
# TABLE OF CONTENTS

**DEDICATION** ................................................................. iii

**ACKNOWLEDGEMENTS** ...................................................... iv

**LIST OF FIGURES** ........................................................... vii

**SUMMARY** ........................................................................ viii

1 **INTRODUCTION** ............................................................... 1

2 **BACKGROUND** ................................................................. 5

   2.1 2D Classical State Space models ........................................ 6

   2.1.1 Givone–Roesser 2D model ............................................. 6

   2.1.2 Attasi 2D model .......................................................... 7

   2.1.3 Fornasini-Marchesini 2D models .................................... 8

3 **SECOND-ORDER TWO-DIMENSIONAL SYSTEM** .................... 9

   3.1 2O2D Background ......................................................... 9

   3.2 2D Discrete Fourier Transform ......................................... 12

   3.3 Algorithm .................................................................... 14

   3.3.1 Denominator Polynomial ............................................ 15

   3.3.2 Numerator Polynomial ................................................. 16

   3.4 Numerical Examples ..................................................... 16

   3.4.1 Single–Input Single–Output ......................................... 16

   3.4.2 Multiple–Input Single–Output ...................................... 21

   3.4.3 Multiple–Input Multiple–Output .................................. 25

   3.5 Complexity of the Algorithm ......................................... 32

4 **GENERALIZED SECOND–ORDER TWO-DIMENSIONAL SYSTEM** 38

   4.1 2D Discrete Fourier Transform ....................................... 39

   4.2 DFT-Based Algorithm .................................................... 39

   4.2.1 Denominator Polynomial ............................................ 40

   4.2.2 Numerator Polynomial ................................................. 41

   4.3 Example: G2O2D, two-inputs two-outputs, system ................ 42

   4.4 Complexity of the Algorithm ......................................... 47
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2O2D state circuit</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>2O2D time diagram</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>Plot of $a_{i_1,i_2}$ (29)</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>Plot of $F_{i_1,i_2}$ (33)</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>Example 3.4.1</td>
<td>22</td>
</tr>
<tr>
<td>6</td>
<td>Example 3.4.2 - First Input</td>
<td>26</td>
</tr>
<tr>
<td>7</td>
<td>Example 3.4.2 - Second Input</td>
<td>27</td>
</tr>
<tr>
<td>8</td>
<td>Example 3.4.3 - First Input - First Output</td>
<td>33</td>
</tr>
<tr>
<td>9</td>
<td>Example 3.4.3 - First Input - Second Output</td>
<td>34</td>
</tr>
<tr>
<td>10</td>
<td>Example 3.4.3 - Second Input - First Output</td>
<td>35</td>
</tr>
<tr>
<td>11</td>
<td>Example 3.4.3 - Second Input - Second Output</td>
<td>36</td>
</tr>
</tbody>
</table>
SUMMARY

In this thesis the discrete Fourier transform is used for determining the coefficients of the determinantal polynomial and the coefficients of the adjoint polynomial matrix for four systems/models. The main contribution of this thesis include:

- A method for computing the transfer function of a second–order 2D system using the DFT.
- A method for computing the transfer function of a generalized second–order 2D system using the DFT.
- The second–order 2D systems was extended to $k$–order $n$–dimensions.
- The generalized second order 2D system was extended to generalized $k$–order $n$–dimensions.
- A method for computing the transfer function of a $k$–order $n$–dimensional system using the DFT.
- A method for computing the transfer function of a generalized $k$–order $n$–dimensional system using the DFT.
CHAPTER 1

INTRODUCTION

Multidimensional (nD) digital systems have received, in recent decades, considerable research interest because of the advancement of digital technology. One Dimensional (1D) systems find wide applications in speech processing (wired and wireless), audio systems, data and computer communications etc. [1], [10], [30], [31], [32]. Systems that are characterized by two independent variables propagating information in two directions, in systems theory, or by two delay elements $z_1^{-1}, z_2^{-1}$, in circuit theory, require a different approach for the analysis since many 1D techniques can’t be extended to 2D [7], [13], [20]. It is noted that systems with multiple delays are used today in electronic industry. The multiple delay elements are considered to be as “scheduling devices”, implemented with house software (Xilinx™) over MATLAB™ and hardware with the well known D–type Flip Flops [14], [35]. Applications of Multidimensional systems can be found in digital image processing, seismology, geophysics, remote sensing, robotics, medicine, underwater acoustics, metal–rolling, long-wall coal cutting, etc. [10], [16], [20], [21], [25], [26], [34], [36], [37].

Multidimensional systems, like 1D, can be represented with a polynomial transfer function:

$$T_1(z_1, z_2, z_3, \ldots, z_n) = \frac{N(z_1, z_2, z_3, \ldots, z_n)}{D(z_1, z_2, z_3, \ldots, z_n)}$$

or using classical state space like structures, for example:

$$\dot{x}(i_1, i_2, \ldots, i_n) = Ax(i_1, i_2, \ldots, i_n) + bu(i_1, i_2, \ldots, i_n)$$
$$y(i_1, i_2, \ldots, i_n) = c'x(i_1, i_2, \ldots, i_n) + du(i_1, i_2, \ldots, i_n)$$

The classical state equation $\dot{x}(i_1, i_2, \ldots, i_n)$ and output equation $y(i_1, i_2, \ldots, i_n)$, represent the system dynamics and the input-output behavior of the system. The model is linear,
shift invariant and causal [20].

State space based techniques play a very crucial role in the analysis, synthesis and implementation of multidimensional systems.

In this thesis, new system–models for representing $k$–order $n$–dimensional systems are presented for the regular and generalized structures. Using the proposed models, computer implementable algorithms are proposed, using the discrete Fourier transform (DFT), for the computation of the transfer function for the following models:

- **Second–order 2D system**
  
  

- **Generalized second–order 2D system**
  
  

- **$k$–order $n$–D system**
  
Generalized $k$–order $n$–D system


The new system-models are logical extensions for multidimensional systems. The particular models are extension of the 2D Fornasini-Marchesini [11] and the second–order two–dimensional models [3] to $n$-Dimensions and $k$-Order. For more two–dimensional (2D) second–order system structures, the reader can refer to [3] . The proposed extension to $k$–order systems, will increase the order of the numerator and denominator polynomials of the $nD$ transfer function and at the same time will represent systems in more than one dimension. It is noted that the presented model can realize a transfer function with lower matrix-dimensions than the classical models.

1D systems/models with order more that one have been used in the past for Robo$t$ pole placement, Model reduction, Real time block recursive parameter estimation, Leverrier and extension of Fadeevs’s method to polynomial matrices, in electric power analysis and analysis of flexible beams [8], [9], [17], [19], [27], [29].

In this thesis the DFT is used for determining the coefficients of the determinantal polynomial and the coefficients of the adjoint polynomial matrix for four systems/models. The computational speed of the proposed method can be improved by using fast Fourier transform techniques.

The DFT has been used for the evaluation of the transfer function coefficients for linear, generalized (singular), and multidimensional state space systems [2], [4], [22], [33]. Other methods that could have been used are the Leverrier–Fadeeva, and Vandermonde matrices [6], [18], [28].

The main contribution of this thesis include:

• A method for computing the transfer function of a second-order 2D system using the DFT.
• A method for computing the transfer function of a generalized second-order 2D system using the DFT.

• The second-order 2D systems was extended to $k$–order $n$–dimensions.

• The generalized second order 2D system was extended to generalized $k$–order $n$–dimensions.

• A method for computing the transfer function of a $k$–order $n$–dimensional system using the DFT.

• A method for computing the transfer function of a generalized $k$–order $n$–dimensional system using the DFT.

The thesis is organized as follows: Chapter 1 is the introduction. Chapter 2 contains some background information. Chapter 3 explains a method for computing the transfer function of a second-order 2D system using the DFT. Chapter 4 demonstrates a method for computing the transfer function of a generalized second-order 2D system using the DFT. Chapter 5 has the extended $k$–order $n$–dimensional system and a method for computing the transfer function using the DFT. Chapter 6 presents the extended generalized $k$–order $n$–dimensional system and a method for computing the transfer function using the DFT. Chapter 7 comprises the conclusions. Chapter 8 presents future work that can be done.
A 2D discrete signal can be modeled as a function of two independent variables \(x(i_1, i_2)\), defined for all integer values of \(i_1\) and \(i_2\) [25].

A 2D digital system is defined as an operator, \(T\), that transforms an input \(x(i_1, i_2)\) to an output \(y(i_1, i_2)\).

\[
y(i_1, i_2) = T(x(i_1, i_2))
\]

For linearity,

\[
T[ax(i_1, i_2) + by(i_1, i_2)] = aT[x(i_1, i_2)] + bT[y(i_1, i_2)]
\]

\(\forall\ a, b\) arbitrary constants.

The 2D \(z\)-transform of a function \(x(n_1, n_2)\) is defined as:

\[
X(z_1, z_2) = \sum_{i_1=-\infty}^{\infty} \sum_{i_2=-\infty}^{\infty} x(i_1, i_2)z_1^{-i_1}z_2^{-i_2}
\]

where, \(z_1\) and \(z_2\) are complex variables.

For example a 2D system is described by the following difference equation [25]:

\[
\sum_{(k_1, k_2) \in R_a} a(k_1, k_2)y(n_1 - k_1, n_2 - k_2) = \sum_{(k_1, k_2) \in R_b} b(k_1, k_2)u(n_1 - k_1, n_2 - k_2)
\]

where, \(u(n_1, n_2)\), \(y(n_1, n_2)\) are the input and output arrays and \(a(k_1, k_2)\), \(b(k_1, k_2)\) are 2D finite sequences. Applying the 2D \(z\)-transform on both sides of the above equation yields,

\[
H(z_1, z_2) = \frac{Y(z_1, z_2)}{U(z_1, z_2)} = \sum_{(n_1, n_2) \in R_b} \sum_{(n_1, n_2) \in R_a} b(n_1, n_2)z_1^{-n_1}z_2^{-n_2}
\]

\[
\sum_{(n_1, n_2) \in R_a} \sum_{(n_1, n_2) \in R_b} a(n_1, n_2)z_1^{-n_1}z_2^{-n_2}
\]

The above numerator- denominator description \(H(z_1, z_2)\) is a 2D transfer function.
2.1 2D Classical State Space models

In system theory two approaches are used for the analysis and design: the transfer function and the state space. The definition of the 2D transfer function was given and an example from the literature was presented in the previous section. It is noted that the reason for presenting the 2D case is only due to simplicity.

There are the three classical 2D state space/model descriptions:

- Givone–Roesser
- Attasi
- Fornasini–Marchesini

2.1.1 Givone–Roesser 2D model

In 1972, Givone–Roesser introduced the first spatial state space system for linear iterative circuits. Iterative circuits are a combination of individual cells [15]. The updating state equation and the output equations are:

\[
\begin{bmatrix}
  x^h_{n_1}(i_1, i_2 + 1) \\
  x^v_{n_2}(i_1, i_2)
\end{bmatrix} =
\begin{bmatrix}
  A_{11} & A_{12} \\
  A_{21} & A_{22}
\end{bmatrix}
\begin{bmatrix}
  x^h_{n_1}(i_1, i_2) \\
  x^v_{n_2}(i_1, i_2)
\end{bmatrix} +
\begin{bmatrix}
  B_1 \\
  B_2
\end{bmatrix} u(i, j)
\]

\[y(i, j) =
\begin{bmatrix}
  C'_1 & C'_2
\end{bmatrix}
\begin{bmatrix}
  x^h_{n_1}(i_1, i_2) \\
  x^v_{n_2}(i_1, i_2)
\end{bmatrix} + du(i_1, i_2)
\]

where,

\[x^h(i, j) \in R^{n_1}.. \text{Horizontal State Vector}\]

\[x^v(i, j) \in R^{n_2}.. \text{Vertical State Vector}\]

\[y (i, j) \in R^p.. \text{Output Vector}\]

\[u (i, j) \in R^m.. \text{Input Vector}\]
In a compact form the above state space model (1) can be written as:

\[
\begin{align*}
\dot{x}(i_1, i_2) &= Ax(i_1, i_2) + Bu(i_1, i_2) \\
y(i, j) &= C'x(i_1, i_2) + du(i_1, i_2)
\end{align*}
\]  

(2)

where,

\[
\begin{align*}
\dot{x}(i_1, i_2) &= \begin{bmatrix} x_h(i_1 + 1, i_2) \\
x_v(i_1, i_2 + 1) \end{bmatrix} \in \mathbb{R}^{n_1+n_2} \\
x(i_1, i_2) &= \begin{bmatrix} x_h(i_1, i_2) \\
x_v(i_1, i_2) \end{bmatrix} \in \mathbb{R}^{n_1+n_2}
\end{align*}
\]

and

\[
A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, C = \begin{bmatrix} C_1 & C_2 \end{bmatrix}
\]

\[
A_{11}, A_{12}, A_{21}, A_{22}, B_1, B_2, C_1, C_2, \text{ are real matrices of appropriate dimensions and } d \text{ is a scalar.}
\]

Applying the 2D \(z\)-transform on (2), with zero initial conditions, the corresponding 2D transfer function takes the following form:

\[
H_{gr}(z_1, z_2) = C'[Z - A]^{-1}B + d
\]  

(3)

where, \(Z = z_1I_n \oplus z_2I_n\), with \(\oplus\) denoting the direct sum.

2.1.2 Attasi 2D model

In 1973, S. Attasi proposed the following 2D model [5]:

\[
\begin{align*}
x(i_1 + 1, i_2 + 1) &= A_1x(i_1 + 1, j) + A_2x(i_1, i_2 + 1) + A_0x(i_1, i_2) + Bu(i_1, i_2) \\
y(i, j) &= C'x(i_1, i_2)
\end{align*}
\]  

(4)
Applying the 2D $z$-transform the following 2D transfer function is derived,

$$H_a(z_1, z_2) = C'[z_1 z_2 \mathcal{I} - z_1 A_1 - z_2 A_2 - A_0]^{-1} B$$  \hfill (5)

### 2.1.3 Fornasini-Marchesini 2D models

In 1976 and 1978, Fornasini–Marchesini proposed the following 2D models:

**Fornasini–Marchesini First 2D state space model** [11]:

$$x(i_1 + 1, i_2 + 1) = A_0 x(i_1, i_2) + A_1 x(i_1 + 1, i_2) + A_2 x(i_1, i_2 + 1) + B u(i_1, i_2)$$  \newline
$$y(i_1, i_2) = C' x(i_1, i_2)$$  \hfill (6)

**Fornasini–Marchesini First 2D transfer function:**

$$H_{fm1}(z_1, z_2) = c'[z_1 z_2 \mathcal{I} - z_1 A_1 - z_2 A_2 - A_0]^{-1} B$$  \hfill (7)

**Fornasini–Marchesini Second 2D state space model** [12]:

$$x(i_1 + 1, i_2 + 1) = A_1 x(i_1 + 1, i_2) + A_2 x(i_1, i_2 + 1) + B_1 u(i_1 + 1, i_2) + B_2 u(i_1, i_2 + 1)$$  \newline
$$y(i, j) = C' x(i_1, i_2)$$  \hfill (8)

**Fornasini–Marchesini Second 2D transfer function:**

$$H_{fm1}(z_1, z_2) = c'[z_1 z_2 \mathcal{I} - z_1 A_1 - z_2 A_2]^{-1} B_1 z_1 + B_2 z_2$$  \hfill (9)

It is noted that the Fornasini–Marchesini models are based on the algebraic viewpoint of Nerode equivalence [7].

In the following section the mathematical representation of the new model is analyzed by defining the properties of the model. This model is of second–order and two dimensions.
A second–order two–dimensional (SO2D) system has the following structure [3]:

\[
x(i_1 + 2, i_2 + 2) = A_0 x(i_1 + 1, i_2 + 1) + A_1 x(i_1 + 1, i_2) + A_2 x(i_1, i_2 + 1) \\
+ B_1 u(i_1 + 1, i_2) + B_2 u(i_1, i_2 + 1)
\]

\[
y(i_1, i_2) = C x(i_1, i_2)
\]

where, \( x(i_1, i_2) \in \mathbb{R}^n \), \( u(i_1, i_2) \in \mathbb{R}^m \), \( y(i_1, i_2) \in \mathbb{R}^p \), \( i_1, i_2 \) are integer–valued vertical and horizontal coordinates respectively, \( x(i_1, i_2) \) is the local vector at \((i_1, i_2)\), \( u(i_1, i_2) \) is the input vector and \( y(i_1, i_2) \) is the output vector. \( A_k \), for \( k = 0, 1, 2 \) and \( B_1, B_2, C \), are real matrices of appropriate dimensions denoting the characteristics of the SO2D system that can also be represented by a 2D transfer function, as in the regular 2D systems [12]. It is noted that this particular SO2D system (10) is an extension of the regular 2D Fornasini–Marchesini model [12] to cover systems of second–order. The “state” circuit for the SO2D system (10) is depicted in Figure 1. For more 2D second–order structures the reader can refer to [3].

Applying the 2D \( z_i \), \( i = 1, 2 \) transform to system (10), with zero initial conditions, the transfer function is found to be:

\[
T(z_1, z_2) = C \left[ I z_1^2 z_2^2 - A_0 z_1 z_2 - A_1 z_1 - A_2 z_2 \right]^{-1} \cdot [B_1 z_1 + B_2 z_2]
\]

\[
3.1 \textbf{2O2D Background}
\]

To help the reader consider the SO2D model (10) with some numerical constant coefficients, for example:
Figure 1: 2O2D state circuit
\[ x(i_1 + 2, i_2 + 2) = x(i_1 + 1, i_2 + 1) + 2x(i_1 + 1, i_2) + 3x(i_1, i_2 + 1) + u(i_1 + 1, i_2) + 2u(i_1, i_2 + 1) \]

\[ y(i_1, i_2) = x(i_1, i_2) \tag{12} \]

It’s transfer function becomes

\[ T(z_1, z_2) = \frac{z_1 + 2z_2}{z_1^2 z_2 - z_1 z_2 - 2z_1 + 3z_2}. \]

In comparison, with the “equivalent” classical 2D Fornasini–Marchesini model [12],

\[ x(i_1 + 1, i_2 + 1) = 2x(i_1 + 1, i_2) + 3x(i_1, i_2 + 1) + u(i_1 + 1, i_2) + 2u(i_1, i_2 + 1) \]

\[ y(i_1, i_2) = x(i_1, i_2) \tag{13} \]

The transfer function,

\[ T(z_1, z_2) = C [Iz_1 z_2 - A_1 z_1 - A_2 z_2]^{-1} \cdot (B_1 z_1 + B_2 z_2) \tag{14} \]

or

\[ T(z_1, z_2) = \frac{z_1 + 2z_2}{z_1 z_2 - z_1 z_2 - 2z_1 + 3z_2}. \tag{15} \]

The Figure 2 depicts the relation among the “state” and any point \((i_1, i_2)\). Also the “state” at the neighborhood points \((i_1 - 1, i_2)\), \((i_1 - 1, i_2 - 1)\) and \((i_1, i_2 - 1)\) and the values of the boundary conditions \(x(i_1, 0)\) and \(x(0, i_2)\).

It is clear that the value of a “state” at any point depends on the entire neighbor “state” values of all the neighbor points. The boundary conditions too are simple. If for example we assume zero inputs, then the “state” at various \(x(.,.)\) points are given by:

\[ x(2, 2) = A_0 x(1, 1) + A_1 x(1, 0) + A_2 x(0, 1) \]

\[ x(2, 3) = A_0 x(1, 2) + A_1 x(1, 1) + A_2 x(0, 2) \]
\[x(2, 4) = A_0 x(1, 3) + A_1 x(1, 2) + A_2 x(0, 3)\]
\[x(2, 5) = A_0 x(1, 4) + A_1 x(1, 3) + A_2 x(0, 4)\]
\[x(3, 2) = A_0 x(2, 1) + A_1 x(2, 0) + A_2 x(1, 1)\]
\[x(3, 3) = A_0 x(2, 2) + A_1 x(2, 1) + A_2 x(1, 2)\]
\[x(3, 4) = A_0 x(2, 3) + A_1 x(2, 2) + A_2 x(1, 3)\]
\[x(3, 5) = A_0 x(2, 4) + A_1 x(2, 3) + A_2 x(1, 4)\]
\[x(4, 2) = A_0 x(3, 1) + A_1 x(3, 0) + A_2 x(2, 1)\]
\[x(4, 3) = A_0 x(3, 2) + A_1 x(3, 1) + A_2 x(2, 2)\]
\[x(4, 4) = A_0 x(3, 3) + A_1 x(3, 2) + A_2 x(2, 3)\]
\[x(4, 5) = A_0 x(3, 4) + A_1 x(3, 3) + A_2 x(2, 4)\]
\[x(5, 2) = A_0 x(4, 1) + A_1 x(4, 0) + A_2 x(3, 1)\]
\[x(5, 3) = A_0 x(4, 2) + A_1 x(4, 1) + A_2 x(3, 2)\]
\[x(5, 4) = A_0 x(4, 3) + A_1 x(4, 2) + A_2 x(3, 3)\]
\[x(5, 5) = A_0 x(4, 4) + A_1 x(4, 3) + A_2 x(3, 4)\]

The Figure 2 clearly shows the above analysis.

In the following section an interpolative approach is developed for determining the transfer function \(T(z_1, z_2)\), given the matrices \(A_k, k = 0, 1, 2\) and \(B_1, B_2, C\) using the 2D DFT.

For the sake of completeness a brief description of the 2D DFT follows.

### 3.2 2D Discrete Fourier Transform

Consider the finite sequences \(X(k_1, k_2)\) and \(\tilde{X}(r_1, r_2)\), \(k_i, r_i = 0, \ldots, M_i, \forall i = 1, 2\). In order for the sequences \(X(k_1, k_2)\) and \(\tilde{X}(r_1, r_2)\), to constitute a 2D DFT pair the following relations should hold [10]:

\[\tilde{X}(r_1, r_2) = \sum_{k_1=0}^{M_1-1} \sum_{k_2=0}^{M_2-1} X(k_1, k_2) e^{-j 2\pi (r_1 k_1 / M_1 + r_2 k_2 / M_2)}\]
Figure 2: 2O2D time diagram
\[ \tilde{X}(r_1, r_2) = \sum_{k_1=0}^{M_1} \sum_{k_2=0}^{M_2} X(k_1, k_2) W_1^{-k_1 r_1} W_2^{-k_2 r_2} \] (16)

\[ X(k_1, k_2) = \frac{1}{R} \sum_{r_1=0}^{M_1} \sum_{r_2=0}^{M_2} \tilde{X}(r_1, r_2) W_1^{k_1 r_1} W_2^{k_2 r_2} \] (17)

where,

\[ R = (M_1 + 1)(M_2 + 1) \] (18)

\[ W_i = e^{(2\pi j)/(M_i+1)}, \ i = 1, 2 \] (19)

\( X, \tilde{X} \) are discrete argument matrix valued functions, with dimensions \( p \times m \).

In the following section an interpolative approach is developed for determining the transfer function \( T(s) \), given the matrices \( A_i, i = 0, 1, 2 \) and \( B_1, B_2, C \), using the 2D DFT.

### 3.3 Algorithm

The transfer function of a SO2D system (10) is,

\[ T(z_1, z_2) = \frac{N(z_1, z_2)}{d(z_1, z_2)} \] (20)

where,

\[ N(z_1, z_2) = C \ \text{adj} \ [I z_1^2 z_2^2 - A_0 z_1 z_2 - A_1 z_1 - A_2 z_2] \cdot (B_1 z_1 + B_2 z_2) \] (21)

\[ d(z_1, z_2) = \det [I z_1^2 z_2^2 - A_0 z_1 z_2 - A_1 z_1 - A_2 z_2] \] (22)

Equations (21) and (22) can be written in polynomial form as follows:

\[ N(z_1, z_2) = \sum_{\lambda_1=0}^{n_{max}} \sum_{\lambda_2=0}^{n_{max}} P_{\lambda_1, \lambda_2} z_1^{\lambda_1} z_2^{\lambda_2} \] (23)

with, \( n_{max}^P := \max((2\lambda_1 - 1), (2\lambda_2 - 1)) \). The numerator coefficients \( P_{\lambda_1, \lambda_2} \) are matrices with dimensions \( (p \times m) \).
\[ d(z_1, z_2) = \sum_{\lambda_1=0}^{n_{\text{max}}^1} \sum_{\lambda_2=0}^{n_{\text{max}}^2} q_{\lambda_1, \lambda_2} z_1^{\lambda_1} z_2^{\lambda_2} \]  

(24)

where, \( n_{\text{max}} := \max(2\lambda_1, 2\lambda_2) \). The denominator coefficients \( q_{\lambda_1, \lambda_2} \) are scalars.

The numerator polynomial matrix \( N(z_1, z_2) \) and the denominator polynomial \( d(z_1, z_2) \) can be numerically computed at \( R = (r + 1)^2 \), points, equally spaced on the unit 2D disc. The \( R \) points are chosen as \( (z_1, z_2) = [v(i_1), v(i_2)] \), \( i_1, i_2 = 0, \ldots, r \), with \( r = 2\lambda \), according to definition as:

\[ v_1(i) = v_2(i) = W^{-i}, \quad \forall \ i = 0, \ldots, r. \]  

(25)

where,

\[ W_i = e^{(2\pi j)/(r+1)}, \quad i = 1, 2 \]  

(26)

The values of the transfer function (20) at the \( R \) points are the corresponding 2D DFT coefficients.

### 3.3.1 Denominator Polynomial

To evaluate the denominator coefficients \( q_{\lambda_1, \lambda_2} \), define,

\[ a_{i_1, i_2} = \det \begin{bmatrix} I v_1^2(i_1) v_2^2(i_2) - A_0 v_1(i_1) v_2(i_2) - A_1 v_1(i_1) + A_2 v_2(i_2) \end{bmatrix} \]  

(27)

Therefore using equations (24) and (27), \( a_{i_1, i_2} \) can be defined as,

\[ a_{i_1, i_2} = d[v_1(i_1), v_2(i_2)] \]  

(28)

Provided that at least one of \( a_{i_1, i_2} \neq 0 \).

Equations (24), (25) and (28) yield

\[ a_{i_1, i_2} = \sum_{\lambda_1=0}^{r} \sum_{\lambda_2=0}^{r} q_{\lambda_1, \lambda_2} W^{-(i_1 \lambda_1 + i_2 \lambda_2)} \]  

(29)

In the above equation (29) it is obvious that \( [a_{i_1, i_2}] \) and \( [q_{\lambda_1, \lambda_2}] \) form a 2D DFT pair. Therefore the coefficients \( [q_{\lambda_1, \lambda_2}] \) can be computed using the inverse 2D DFT, as follows:
\[ g_{\lambda_1, \lambda_2} = \frac{1}{R} \sum_{i_1=0}^{r-1} \sum_{i_2=0}^{r-1} a_{i_1, i_2} W^{(i_1 \lambda_1 + i_2 \lambda_2)} \] (30)

### 3.3.2 Numerator Polynomial

To evaluate the numerator matrix polynomial \( P_{\lambda_1, \lambda_2} \), define

\[
F_{i_1, i_2} = C \text{adj} \left[ I v_1^2(i_1) v_2^2(i_2) - A_0 v_1(i_1) v_2(i_2) - A_1 v_1(i_1) - A_2 v_2(i_2) \right] 
\times [B_1 v_1(i_1) + B_2 v_2(i_2)]
\] (31)

Using equations (23) and (31), \( F_{i_1, i_2} \) can be defined as,

\[ F_{i_1, i_2} = N[v_1(i_1), v_2(i_2)] \] (32)

Equations (23), (25) and (32) yield

\[
F_{i_1, i_2} = \sum_{\lambda_1=0}^{r-1} \sum_{\lambda_2=0}^{r-1} P_{\lambda_1, \lambda_2} W^{-(i_1 \lambda_1 + i_2 \lambda_2)}
\] (33)

In the above equation (29), \([ F_{i_1, i_2}, [P_{\lambda_1, \lambda_2}] \) form a 2D DFT pair. Therefore the coefficients \( P_{\lambda_1, \lambda_2} \) can be computed, using the inverse 2D DFT, as follows:

\[
P_{\lambda_1, \lambda_2} = \frac{1}{R} \sum_{i_1=0}^{r-1} \sum_{i_2=0}^{r-1} F_{i_1, i_2} W^{(i_1 \lambda_1 + i_2 \lambda_2)}
\] (34)

Three salient examples, simple yet illustrative of the theoretical concepts presented in this work, follow below:

### 3.4 Numerical Examples

#### 3.4.1 Single–Input Single–Output

Consider the following single–input single–output 2DSO system:

\[
x(i_1 + 2, i_2 + 2) = A_0 x(i_1 + 1, i_2 + 1) + A_1 x(i_1 + 1, i_2) + A_2 x(i_1, i_2 + 1) + B_1 u(i_1 + 1, i_2) + B_2 u(i_1, i_2 + 1)
\]

\[ y(i_1, i_2) = C x(i_1, i_2) \] (35)
where,

\[
\begin{align*}
A_0 &= \begin{bmatrix} 1 & 3 \\ 0 & 0 \end{bmatrix}, & A_1 &= \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix}, & A_2 &= \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \\
B_1 &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}, & B_2 &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
C &= \begin{bmatrix} 0 & 1 \end{bmatrix}
\end{align*}
\]

Since \( \lambda = 2 \), the \( r = 2 \cdot \lambda = 4 \). Therefore \( R = (r + 1)^2 = 25 \). The direct application of the proposed algorithm yields:

Using (29), the \( a_{i_1i_2} \) coefficients are,

\[
\begin{align*}
a_{00} &= -6.0000 + 0.0000j, & a_{01} &= 3.0451 + 2.4899j \\
a_{02} &= -2.5451 + 0.2245j, & a_{03} &= -2.5451 - 0.2245j \\
a_{04} &= 3.0451 - 2.4899j, & a_{10} &= -0.3090 + 4.7553j \\
a_{11} &= 1.9271 + 3.3022j, & a_{12} &= 0.1180 - 2.7144j \\
a_{13} &= 3.3090 + 1.4001j, & a_{14} &= -1.0000 - 3.8042j \\
a_{20} &= 0.8090 + 2.9389j, & a_{21} &= 2.1910 - 4.3920j \\
a_{22} &= -1.4271 - 3.2164j, & a_{23} &= -1.0000 - 2.3511j \\
a_{24} &= -2.1180 + 2.2654j, & a_{30} &= 0.8090 - 2.9389j \\
a_{31} &= -2.1180 - 2.2654j, & a_{32} &= -1.0000 + 2.3511j \\
a_{33} &= -1.4271 + 3.2164j, & a_{34} &= 2.1910 + 4.3920j \\
a_{40} &= -0.3090 - 4.7553j, & a_{41} &= -1.0000 + 3.8042j \\
a_{42} &= 3.3090 - 1.4001j, & a_{43} &= 0.1180 + 2.7144j \\
a_{44} &= 1.9271 - 3.3022j
\end{align*}
\]

The Figure 3 shows the plot of \( a_{i_1i_2} \) (29), using MATLAB\textsuperscript{TM}, given in Appendix-A.

Using (33), the \( F_{i_1i_2} \) coefficients are,
\[
F_{00} = 2.0000 + 0.0000j \quad F_{01} = 0.6180 - 0.7265j
\]
\[
F_{02} = -1.6180 - 3.0777j \quad F_{03} = -1.6180 + 3.0777j
\]
\[
F_{04} = 0.6180 + 0.7265j \quad F_{10} = -0.5000 - 1.5388j
\]
\[
F_{11} = 0.1910 - 1.7634j \quad F_{12} = -3.4271 + 1.7634j
\]
\[
F_{13} = 1.7361 + 1.5388j \quad F_{14} = 2.0000 + 0.0000j
\]
\[
F_{20} = -0.5000 + 0.3633j \quad F_{21} = -2.7361 - 0.3633j
\]
\[
F_{22} = 1.3090 + 2.8532j \quad F_{23} = 2.0000 + 0.0000j
\]
\[
F_{24} = -0.0729 - 2.8532j \quad F_{30} = -0.5000 - 0.3633j
\]
\[
F_{31} = -0.0729 + 2.8532j \quad F_{32} = 2.0000 - 0.0000j
\]
\[
F_{33} = 1.3090 - 2.8532j \quad F_{34} = -2.7361 + 0.3633j
\]
\[
F_{40} = -0.5000 + 1.5388j \quad F_{41} = 2.0000 - 0.0000j
\]
\[
F_{42} = 1.7361 - 1.5388j \quad F_{43} = 3.4271 - 1.7634j
\]
\[
F_{44} = 0.1910 + 1.7634j
\]

The Figure 3 shows the plot of $F_{i_1,i_2}$ (33), using the software package $MATLAB^{TM}$, given in Appendix-A.

Using (30), the denominator coefficients are,

\[
\begin{bmatrix}
q_{00} & q_{01} & q_{02} & q_{03} & q_{04} \\
q_{10} & q_{11} & q_{12} & q_{13} & q_{14} \\
q_{20} & q_{21} & q_{22} & q_{23} & q_{24} \\
q_{30} & q_{31} & q_{32} & q_{33} & q_{34} \\
q_{40} & q_{41} & q_{42} & q_{43} & q_{44}
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & -1 & 0 & 0 \\
0 & -2 & -2 & 0 & 0 \\
-1 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

Using (34), the numerator matrix polynomials are,

\[
\begin{bmatrix}
P_{00} & P_{01} & P_{02} & P_{03} & P_{04} \\
P_{10} & P_{11} & P_{12} & P_{13} & P_{14} \\
P_{20} & P_{21} & P_{22} & P_{23} & P_{24} \\
P_{30} & P_{31} & P_{32} & P_{33} & P_{34} \\
P_{40} & P_{41} & P_{42} & P_{43} & P_{44}
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]
Figure 3: Plot of $a_{i_1,i_2}$ (29)
Figure 4: Plot of $F_{i_1,i_2}$ (33)
Once the denominator and the adjoint matrix coefficients have been computed, the transfer function $T(z_1, z_2)$ is determined as,

$$T(z_1, z_2) = \frac{P_{23}z_1^2z_2^3 + P_{12}z_1z_2^2 + P_{11}z_1z_2}{q_{44}z_1^4z_2^4 + q_{33}z_1^3z_2^3 + q_{23}z_1^2z_2^2 + q_{21}z_1z_2^2 + q_{20}z_1^2 + q_{12}z_1z_2 + q_{11}z_1 + q_{02}z_2^2}$$

or

$$T(z_1, z_2) = \frac{z_1^2z_2^3 - z_1z_2^2 + z_1z_2}{z_1^4z_2^4 - z_1^3z_2^3 - z_1^2z_2^2 + z_1^2z_2 - z_1^2 - 2z_1z_2^2 - 2z_1z_2 - z_2^2}$$  \hspace{1cm} (36)$$

Figure 5 shows the mesh, contour, surface and z plots of the above 2D transfer function (36). The MATLAB$^\text{TM}$ code is given in the Appendix B.

The correctness of the above result (36) can easily be verified using (11), with $k = n = 2,$

$$T(z_1, z_2) = C [Iz_1^2z_2^2 - A_0z_1z_2 - A_1z_1 - A_2z_2]^{-1} \cdot (B_1z_1 + B_2z_2)$$

### 3.4.2 Multiple–Input Single–Output

Consider the following two–input single–output 2DSO system:

$$x(i_1 + 2, i_2 + 2) = A_0 x(i_1 + 1, i_2 + 1) + A_1 x(i_1 + 1, i_2) + A_2 x(i_1, i_2 + 1) + B_1 u(i_1 + 1, i_2) + B_2 u(i_1, i_2 + 1)$$  \hspace{1cm} (37)$$

$$y(i_1, i_2) = C x(i_1, i_2)$$

where,

$$A_0 = \begin{bmatrix} 1 & 3 \\ 0 & 0 \end{bmatrix}, \quad A_1 = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} 1 & 2 \\ -3 & 0 \end{bmatrix}, B_2 = \begin{bmatrix} 3 & 1 \\ 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$
Figure 5: Example 3.4.1
Since $\lambda = 2$, the $r = 2 \cdot \lambda = 4$. Therefore $R = (r + 1)^2 = 25$. The direct application of the proposed algorithm yields:

The coefficients (29) $a_{i_1i_2} \forall i_1, i_2 = 0, ..., 4$ are the same as in the single-input single-output Example 3.4.1

Using (33), the $F_{i_1i_2}$ coefficients are,

\[
\begin{align*}
F_{00} &= \begin{bmatrix} -19.0000 & 2.0000 \end{bmatrix} \\
F_{01} &= \begin{bmatrix} -9.7533 + 18.1558j & -7.4721 - 0.4490j \end{bmatrix} \\
F_{02} &= \begin{bmatrix} 9.2533 + 4.6493j & 1.4721 + 4.9798j \end{bmatrix} \\
F_{03} &= \begin{bmatrix} 9.2533 - 4.6493j & 1.4721 - 4.9798j \end{bmatrix} \\
F_{04} &= \begin{bmatrix} -9.7533 - 18.1558j & -7.4721 + 0.4490j \end{bmatrix} \\
F_{10} &= \begin{bmatrix} 2.1180 + 13.1230j & -0.5000 + 0.8123j \end{bmatrix} \\
F_{11} &= \begin{bmatrix} 22.6074 + 2.9389j & 3.8090 + 4.1145j \end{bmatrix} \\
F_{12} &= \begin{bmatrix} 1.9549 - 12.0005j & 3.8090 - 0.5878j \end{bmatrix} \\
F_{13} &= \begin{bmatrix} -7.6180 - 3.3552j - 0.5000 - 1.5388j \end{bmatrix} \\
F_{14} &= \begin{bmatrix} -2.8820 + 11.0494j & 1.4721 + 3.0777j \end{bmatrix} \\
F_{20} &= \begin{bmatrix} -0.1180 - 6.3471j & -0.5000 - 3.4410j \end{bmatrix} \\
F_{21} &= \begin{bmatrix} -5.3820 - 7.3309j & -0.5000 + 0.3633j \end{bmatrix} \\
F_{22} &= \begin{bmatrix} -3.1074 - 4.7553j & 2.6910 - 6.6574j \end{bmatrix} \\
F_{23} &= \begin{bmatrix} -5.1180 + 5.5146j & -7.4721 - 0.7265j \end{bmatrix} \\
F_{24} &= \begin{bmatrix} 7.5451 - 6.1024j & 2.6910 + 0.9511j \end{bmatrix} \\
F_{30} &= \begin{bmatrix} -0.1180 + 6.3471j & -0.5000 + 3.4410j \end{bmatrix} \\
F_{31} &= \begin{bmatrix} 7.5451 + 6.1024j & 2.6910 - 0.9511j \end{bmatrix} \\
F_{32} &= \begin{bmatrix} -5.1180 - 5.5146j & -7.4721 + 0.7265j \end{bmatrix}
\end{align*}
\]
\[
F_{33} = \begin{bmatrix} -3.1074 + 4.7553j & 2.6910 + 6.6574j \end{bmatrix} \\
F_{34} = \begin{bmatrix} -5.3820 + 7.3309j & -0.5000 - 0.3633j \end{bmatrix} \\
F_{40} = \begin{bmatrix} 2.1180 - 13.1230j & -0.5000 - 0.8123j \end{bmatrix} \\
F_{41} = \begin{bmatrix} -2.8820 - 11.0494j & 1.4721 - 3.0777j \end{bmatrix} \\
F_{42} = \begin{bmatrix} -7.6180 + 3.3552j & -0.5000 + 1.5388j \end{bmatrix} \\
F_{43} = \begin{bmatrix} 1.9549 + 12.0005j & 3.8090 + 0.5878j \end{bmatrix} \\
F_{44} = \begin{bmatrix} 22.6074 - 2.9389j & 3.8090 - 4.1145j \end{bmatrix}
\]

Using (30), the denominator coefficients are,

\[
\begin{bmatrix}
q_{00} & q_{01} & q_{02} & q_{03} & q_{04} \\
q_{10} & q_{11} & q_{12} & q_{13} & q_{14} \\
q_{20} & q_{21} & q_{22} & q_{23} & q_{24} \\
q_{30} & q_{31} & q_{32} & q_{33} & q_{34} \\
q_{40} & q_{41} & q_{42} & q_{43} & q_{44}
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & -1 & 0 & 0 \\
0 & -2 & -2 & 0 & 0 \\
-1 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

Using (34), the numerator matrix polynomials are,

\[
\begin{bmatrix}
P_{00} & P_{01} & P_{02} & P_{03} & P_{04} \\
P_{10} & P_{11} & P_{12} & P_{13} & P_{14} \\
P_{20} & P_{21} & P_{22} & P_{23} & P_{24} \\
P_{30} & P_{31} & P_{32} & P_{33} & P_{34} \\
P_{40} & P_{41} & P_{42} & P_{43} & P_{44}
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & -3 & 0 & 0 \\
0 & -7 & -2 & 0 & 0 \\
-4 & -2 & -9 & 0 & 0 \\
0 & 0 & 1 & 2 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

Once the denominator and the adjoint matrix coefficients have been computed, the
transfer function \( T(z_1, z_2) \) is determined as,

\[
T(z_1, z_2) = \frac{P_{02}z_2^2 + P_{11}z_1z_2 + P_{12}z_1^2z_2^2 + P_{20}z_1^2 + P_{21}z_1^2z_2 + P_{23}z_1^2z_2^2 + P_{32}z_1^2z_2^3}{q_{02}z_2^2 + q_{11}z_1z_2 + q_{12}z_1^2z_2^2 + q_{20}z_1^2 + q_{21}z_1^2z_2 + q_{23}z_1^2z_2^2 + q_{32}z_1^2z_2^3 + q_{44}z_1^2z_2^4}
\]

or

or
\[ T(z_1, z_2) = \frac{P_{02}z_2^2 + P_{11}z_1z_2 + P_{12}z_1^2 + P_{20}z_2^2 + P_{21}z_1^2z_2 + P_{23}z_1^3z_2^3 + P_{32}z_1^3z_2^4}{-z_2^2 - 2z_1z_2 - 2z_1^2 - z_1^2 + z_1z_2 - z_1^2 - \frac{z_1z_2}{z_2} + \frac{z_1z_2}{z_2}} \]

Where,

\[ P_{02} = \begin{bmatrix} -3 & 0 \end{bmatrix}, \quad P_{11} = \begin{bmatrix} -7 & -2 \end{bmatrix} \]
\[ P_{12} = \begin{bmatrix} 0 & 3 \end{bmatrix}, \quad P_{20} = \begin{bmatrix} -4 & -2 \end{bmatrix} \]
\[ P_{21} = \begin{bmatrix} -9 & 0 \end{bmatrix}, \quad P_{23} = \begin{bmatrix} 3 & 1 \end{bmatrix} \]
\[ P_{32} = \begin{bmatrix} 1 & 2 \end{bmatrix} \]

or

\[ T(z_1, z_2) = \frac{1}{-z_2^2 - 2z_1z_2 - 2z_1^2 - z_1^2 + z_1z_2 - z_1^2 - \frac{z_1z_2}{z_2} + \frac{z_1z_2}{z_2}} \begin{bmatrix} \alpha_1 & \alpha_2 \end{bmatrix} \]

(38)

Where,

\[ \alpha_1 = -3z_2^2 - 7z_1z_2 - 4z_1^2 - 9z_1^2z_2 + 3z_1^2z_2^3 + z_1^3z_2^2 \]
\[ \alpha_2 = -2z_1z_2 + 3z_1z_2^3 - 2z_1^2 + z_1^2z_2^3 + 2z_1^3z_2^2 \]

Figures 6 and 7 show the mesh, contour, surface and z plots of the above 2D transfer function (38). The MATLAB\textsuperscript{TM} code is given in the Appendix B.

The correctness of the above result (38 can easily be verified using (11) with \( k = n = 2 \),

\[ T(z_1, z_2) = C [Iz_1^2z_2^2 - A_0z_1z_2 - A_1z_1 - A_2z_2]^{-1} \cdot (B_1z_1 + B_2z_2) \]

3.4.3 Multiple–Input Multiple–Output

Consider the following multivariable (two–input two–output) 2DSO system:

\[ x(i_1 + 2, i_2 + 2) = A_0^1x(i_1 + 1, i_2 + 1) + A_1x(i_1 + 1, i_2) + A_2x(i_1, i_2 + 1) \]
Figure 6: Example 3.4.2 - First Input
Figure 7: Example 3.4.2 - Second Input
\[ + \mathbf{B}_1 u(i_1 + 1, i_2) + \mathbf{B}_2 u(i_1, i_2 + 1) \]

\[ y(i_1, i_2) = \mathbf{C} x(i_1, i_2) \quad (39) \]

where,

\[
\mathbf{A}_0 = \begin{bmatrix} 1 & 3 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{A}_1 = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{A}_2 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}
\]

\[
\mathbf{B}_1 = \begin{bmatrix} 1 & 2 \\ -3 & 0 \end{bmatrix}, \quad \mathbf{B}_2 = \begin{bmatrix} 3 & 1 \\ 0 & 1 \end{bmatrix}
\]

\[
\mathbf{C} = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix}
\]

Since \( \gamma = 2 \), the \( r = n \times \gamma = 4 \). Therefore \( R = (r + 1)^2 = 25 \). The direct application of the proposed algorithm yields:

The coefficients \( a_{i_1i_2} \forall i_1, i_2 = 0, \ldots, 4 \) are the same as in the single-input single-output Example 3.4.1.

Using (33), the \( F_{i_1i_2} \) coefficients are,

\[
\mathbf{F}_{00} = \begin{bmatrix} 1.0000 & 4.0000 \\ -37.0000 & 8.0000 \end{bmatrix}
\]

\[
\mathbf{F}_{01} = \begin{bmatrix} -1.7639 - 3.8042i & 0.1180 - 2.2654i \\ -21.2705 + 32.5074i & -14.8262 - 3.1634i \end{bmatrix}
\]

\[
\mathbf{F}_{02} = \begin{bmatrix} -6.2361 - 2.3511j & -2.1180 - 2.7144j \\ 12.2705 + 6.9474j & 0.8262 + 7.2452j \end{bmatrix}
\]

\[
\mathbf{F}_{03} = \begin{bmatrix} -6.2361 + 2.3511j & -2.1180 + 2.7144j \\ 12.2705 - 6.9474j & 0.8262 - 7.2452j \end{bmatrix}
\]

\[
\mathbf{F}_{04} = \begin{bmatrix} -1.7639 + 3.8042j & 0.1180 + 2.2654j \\ -21.2705 - 32.5074j & -14.8262 + 3.1634j \end{bmatrix}
\]
\[
F_{10} = \begin{bmatrix}
5.7361 - 2.7144j & 0.8090 - 2.4899j \\
9.9721 + 23.5317j & -0.1910 - 0.8653j
\end{bmatrix}
\]

\[
F_{11} = \begin{bmatrix}
-6.2361 + 1.1756j & -1.4271 - 2.9389j \\
38.9787 + 7.0534j & 6.1910 + 5.2901j
\end{bmatrix}
\]

\[
F_{12} = \begin{bmatrix}
5.8992 + 9.8208j & -3.9271 + 3.3022j \\
9.8090 - 14.1801j & 3.6910 + 2.1266j
\end{bmatrix}
\]

\[
F_{13} = \begin{bmatrix}
5.7361 - 2.9919j & 2.3541 + 1.5388j \\
-9.5000 - 9.7023j & 1.3541 - 1.5388j
\end{bmatrix}
\]

\[
F_{14} = \begin{bmatrix}
1.0000 + 3.5267j & 2.1910 + 0.5878j \\
-4.7639 + 25.6255j & 5.1353 + 6.7432j
\end{bmatrix}
\]

\[
F_{20} = \begin{bmatrix}
1.2639 + 2.2654j & -0.3090 - 0.2245j \\
1.0279 - 10.4289j & -1.3090 - 7.1064j
\end{bmatrix}
\]

\[
F_{21} = \begin{bmatrix}
1.2639 - 5.7921j & -4.3541 - 0.3633j \\
-9.5000 - 20.4540j & -5.3541 + 0.3633j
\end{bmatrix}
\]

\[
F_{22} = \begin{bmatrix}
-1.7639 - 1.9021j & 1.9271 + 4.7553j \\
-7.9787 - 11.4127j & 7.3090 - 8.5595j
\end{bmatrix}
\]

\[
F_{23} = \begin{bmatrix}
1.0000 - 5.7063j & 3.3090 - 0.9511j \\
\end{bmatrix}
\]

\[
F_{24} = \begin{bmatrix}
-6.3992 - 3.1307j & -0.5729 - 3.2164j \\
8.6910 - 15.3354j & 4.8090 - 1.3143j
\end{bmatrix}
\]

\[
F_{30} = \begin{bmatrix}
1.2639 - 2.2654j & -0.3090 + 0.2245j \\
1.0279 + 10.4289j & -1.3090 + 7.1064j
\end{bmatrix}
\]

\[
F_{31} = \begin{bmatrix}
-6.3992 + 3.1307j & -0.5729 + 3.2164j \\
8.6910 + 15.3354j & 4.8090 + 1.3143j
\end{bmatrix}
\]

\[
F_{32} = \begin{bmatrix}
1.0000 + 5.7063j & 3.3090 + 0.9511j \\
-9.2361 - 5.3228j & -11.6353 + 2.4041j
\end{bmatrix}
\]
\[
F_{33} = \begin{bmatrix}
-1.7639 + 1.9021j & 1.9271 - 4.7553j \\
-7.9787 + 11.4127j & 7.3090 + 8.5595j
\end{bmatrix}
\]
\[
F_{34} = \begin{bmatrix}
1.2639 + 5.7921j & -4.3541 + 0.3633j \\
-9.5000 + 20.4540j & -5.3541 - 0.3633j
\end{bmatrix}
\]
\[
F_{40} = \begin{bmatrix}
5.7361 + 2.7144j & 0.8090 + 2.4899j \\
9.9721 - 23.5317j & -0.1910 + 0.8653j
\end{bmatrix}
\]
\[
F_{41} = \begin{bmatrix}
1.0000 - 3.5267j & 2.1910 - 0.5878j \\
-4.7639 - 25.6255j & 5.1353 - 6.7432j
\end{bmatrix}
\]
\[
F_{42} = \begin{bmatrix}
5.7361 + 2.9919j & 2.3541 - 1.5388j \\
-9.5000 + 9.7023j & 1.3541 + 1.5388j
\end{bmatrix}
\]
\[
F_{43} = \begin{bmatrix}
5.8992 - 9.8208j & -3.9271 + 3.3022j \\
9.8090 + 14.1801j & 3.6910 - 2.1266j
\end{bmatrix}
\]
\[
F_{44} = \begin{bmatrix}
-6.2361 - 1.1756j & -1.4271 + 2.9389j \\
38.9787 - 7.0534j & 6.1910 - 5.2901j
\end{bmatrix}
\]

Using (30), the denominator coefficients are,

\[
\begin{bmatrix}
q_{00} & q_{01} & q_{02} & q_{03} & q_{04} \\
q_{10} & q_{11} & q_{12} & q_{13} & q_{14} \\
q_{20} & q_{21} & q_{22} & q_{23} & q_{24} \\
q_{30} & q_{31} & q_{32} & q_{33} & q_{34} \\
q_{40} & q_{41} & q_{42} & q_{43} & q_{44}
\end{bmatrix}
= \begin{bmatrix}
0 & 0 & -1 & 0 & 0 \\
0 & -2 & -2 & 0 & 0 \\
-1 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

Using (34), the numerator matrix polynomials are,

\[
\begin{bmatrix}
P_{00} & P_{01} & P_{02} & P_{03} & P_{04} \\
P_{10} & P_{11} & P_{12} & P_{13} & P_{14} \\
P_{20} & P_{21} & P_{22} & P_{23} & P_{24} \\
P_{30} & P_{31} & P_{32} & P_{33} & P_{34} \\
P_{40} & P_{41} & P_{42} & P_{43} & P_{44}
\end{bmatrix}
\]

30
where the matrices $P_{00}, ..., P_{44}$ are,

\[
\begin{bmatrix}
0 & 0 & \begin{bmatrix} 3 & 1 \\ -3 & 1 \end{bmatrix} & 0 & 0 \\
0 & \begin{bmatrix} 1 & 3 \\ -13 & -1 \end{bmatrix} & 0 & -1 & 0 & 0 \\
-3 & 0 & \begin{bmatrix} 3 & 0 \\ -15 & 0 \end{bmatrix} & 0 & 0 & 0 \\
-11 & -4 & 0 & 0 & 0 & 0 \\
0 & 0 & \begin{bmatrix} -3 & 0 \\ -1 & 4 \end{bmatrix} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix}
\]

Once the denominator and the adjoint matrix coefficients have been computed the transfer function, $T(z_1, z_2)$, of the two–input two–output 2D2O system (39), becomes

\[
T(z_1, z_2) = \frac{P_{02}z_2^2 + P_{11}z_1z_2 + P_{12}z_1^2 + P_{20}z_2^2 + P_{21}z_1^2z_2 + P_{23}z_1^2z_2^3 + P_{32}z_1^3z_2^2}{q_{02}z_2^2 + q_{11}z_1z_2 + q_{12}z_1^2 + q_{20}z_2^2 + q_{21}z_1^2z_2 + q_{23}z_1^2z_2^3 + q_{32}z_1^3z_2^2 + q_{44}z_1^4z_2^4}
\]

or

\[
T(z_1, z_2) = \frac{P_{02}z_2^2 + P_{11}z_1z_2 + P_{12}z_1^2 + P_{20}z_2^2 + P_{21}z_1^2z_2 + P_{23}z_1^2z_2^3 + P_{32}z_1^3z_2^2}{-z_2^2 - 2z_1z_2 - 2z_1^2 - z_1^2 + z_2^2 - z_1^2z_2 - z_2^2 + z_1^3 + z_2^3 - z_1^3z_2 + z_2^3}
\]

Where,

\[
\begin{align*}
P_{02} &= \begin{bmatrix} 3 & 1 \\ -3 & 1 \end{bmatrix}, & P_{11} &= \begin{bmatrix} 1 & 3 \\ -13 & -1 \end{bmatrix}, & P_{12} &= \begin{bmatrix} 0 & -1 \\ 0 & 5 \end{bmatrix} \\
P_{20} &= \begin{bmatrix} -3 & 0 \\ -11 & -4 \end{bmatrix}, & P_{21} &= \begin{bmatrix} 3 & 0 \\ -15 & 0 \end{bmatrix} \\
P_{23} &= \begin{bmatrix} 0 & 1 \\ 6 & 3 \end{bmatrix}, & P_{32} &= \begin{bmatrix} -3 & 0 \\ -1 & 4 \end{bmatrix}
\end{align*}
\]

or
\[ T(z_1, z_2) = \begin{pmatrix} \alpha_1 & \alpha_2 \\ -- & -- \\ \alpha_3 & \alpha_4 \end{pmatrix} \begin{pmatrix} -z_2^2 - 2z_1z_2 - 2z_1^2 - z_1^3 - z_2^3 + z_2^4 \\ -z_1^2 + z_1^3 - z_1^4 + z_2^3 + z_2^4 \end{pmatrix} \] (40)

Where,

\[
\begin{align*}
\alpha_1 &= 3z_2^2 + z_1z_2 - 3z_1^2 + 3z_1z_2^2 - 3z_1^3 z_2^2 \\
\alpha_2 &= z_2^2 + 3z_1z_2 - z_1 z_2^2 + z_1^3 z_2 \\
\alpha_3 &= -3z_2^2 - 13z_1z_2 - 11z_1^2 - 15z_1z_2^2 + 6z_1^2 z_2^3 - z_1^3 z_2^2 \\
\alpha_4 &= z_2^2 - z_1z_2 + 5z_1z_2^2 - 4z_1^2 + 3z_1^3 z_2^2 + 4z_1^3 z_2^2
\end{align*}
\]

Figures 8, 9, 10 and 11 show the mesh, contour, surface and z plots of the above 2D transfer function (40). The \textit{MATLAB} code is given in the Appendix B.

The correctness of the above result (40) can easily be verified using (11), with \( k = n = 2 \),

\[ T(z_1, z_2) = C \left[ I z_1^2 - A_0 z_1 z_2 - A_1 z_1 - A_2 z_2 \right]^{-1} (B_1 z_1 + B_2 z_2). \]

Note–1: It is noted that the structure of the model and the DFT-based, model–to–transfer function, algorithm was illustrated by low order and dimension examples due to simplicity and space requirements.

Note–2: The commas in the subscripts of \( a, F, q, P \), were omitted from the examples for clarification purposes.

### 3.5 Complexity of the Algorithm

The proposed algorithm has two parts. In the first part the matrices \( F_{i_1,i_2} \) and the scalars \( a_{i_1,i_2} \) are evaluated with a cost of \( pn R \lambda^3 \) operations. In the second part the coefficients of \( P_{\lambda_1,\lambda_2} \) and \( q_{\lambda_1,\lambda_2} \) are evaluated using the DFT with a cost of \( pn R^2 + R^2 \) operations. For more efficient computation, especially for high order systems, fast Fourier methods can be used to implement the DFT [31].
Figure 8: Example 3.4.3 - First Input - First Output
Figure 9: Example 3.4.3 - First Input - Second Output
Figure 10: Example 3.4.3 - Second Input - First Output
Figure 11: Example 3.4.3 - Second Input - Second Output
Due to the inherent modularity and the algorithmic structure of the presented method high parallelism is permitted. In this case the computation of each determinant $a_{i_1,j_2}$, (29), and each matrix product $F_{i_1,j_2}$, (33), can be distributed over a number of processing elements, considerably reducing the computation time of the algorithm.
CHAPTER 4

GENERALIZED SECOND–ORDER TWO-DIMENSIONAL SYSTEM

A generalized second–order two-dimensional (G2O2D) system has the following structure [3]:

\[
E x(i_1 + 2, i_2 + 2) = A_0 x(i_1 + 1, i_2 + 1) + A_1 x(i_1 + 1, i_2) + A_2 x(i_1, i_2 + 1) + B_1 x(i_1 + 1, i_2) + B_2 x(i_1, i_2 + 1)
\]

\[
y(i_1, i_2) = C x(i_1, i_2)
\]

where, \( x(i_1, i_2) \in \mathcal{R}^\lambda \), \( u(i_1, i_2) \in \mathcal{R}^m \), \( y(i_1, i_2) \in \mathcal{R}^p \); \( A_k \), for \( k = 1, 2 \) and \( E, B_1, B_2, C \), are real matrices of appropriate dimensions. Matrix \( E \) may be singular.

It is noted that this particular S2O2D system (41) is an extension of the 2D Fornasini–Marchesini model [12] to cover systems of second-order. For more 2D second-order structures the reader can refer to [3]. F.L Lewis has given a survey on 1–D and a review on 2–D generalized or singular or implicit systems [23], [24].

Applying the 2D \( z_i, i = 1, 2 \), transform to system (41), with zero initial conditions, the transfer function is found to be:

\[
T(z_1, z_2) = C \left[ E z_1^2 z_2^2 - A_0 z_1 z_2 - A_1 z_1 - A_2 z_2 \right]^{-1} \cdot (B_1 z_1 + B_2 z_2)
\]

(42)

In the following section an interpolative approach is developed for determining the transfer function \( T(z_1, z_2) \), given the matrices \( A_k, k = 1, 2 \) and \( B_1, B_2, C \) using the 2D DFT. For the sake of completeness a brief description of the 2D DFT follows.
4.1 2D Discrete Fourier Transform

Consider the finite sequences \( X(k_1, k_2) \) and \( \tilde{X}(r_1, r_2) \), \( k_i, r_i = 0, \ldots, M_i \), \( \forall \ i = 1, 2 \). In order for the sequences \( X(k_1, k_2) \) and \( \tilde{X}(r_1, r_2) \), to constitute a 2D DFT pair the following relations should hold [10]:

\[
\tilde{X}(r_1, r_2) = \sum_{k_1=0}^{M_1} \sum_{k_2=0}^{M_2} X(k_1, k_2) W_1^{-k_1 r_1} W_2^{-k_2 r_2} \tag{43}
\]

\[
X(k_1, k_2) = \frac{1}{R} \sum_{r_1=0}^{M_1} \sum_{r_2=0}^{M_2} \tilde{X}(r_1, r_2) W_1^{k_1 r_1} W_2^{k_2 r_2} \tag{44}
\]

where,

\[
R = (M_1 + 1)(M_2 + 1) \tag{45}
\]

\[
W_i = e^{(2\pi j)/(M_i + 1)}, \ i = 1, 2 \tag{46}
\]

\( X, \tilde{X} \) are discrete argument matrix valued functions, with dimensions \( p \times m \).

In the following section an interpolative approach is developed for determining the transfer function \( T(s) \), given the matrices \( A_i, i = 1, 2 \), \( E \), \( B_1 \), \( B_2 \), \( C \), using the 2D DFT.

4.2 DFT-Based Algorithm

The transfer function of the S2O2D system (41) is,

\[
T(z_1, z_2) = \frac{N(z_1, z_2)}{d(z_1, z_2)} \tag{47}
\]

where,

\[
N(z_1, z_2) = C \text{ adj } \begin{bmatrix} E z_1^2 z_2^2 - A_0 z_1 z_2 - A_1 z_1 - A_2 z_2 \end{bmatrix} \times \begin{bmatrix} B_1 z_1 + B_2 z_2 \end{bmatrix} \tag{48}
\]

\[
d(z_1, z_2) = \text{ det } \begin{bmatrix} E z_1^2 z_2^2 - A_0 z_1 z_2 - A_1 z_1 - A_2 z_2 \end{bmatrix} \tag{49}
\]

Equations (48) and (49) can be written in polynomial form as follows:
\[ N(z_1, z_2) = \sum_{\lambda_1=0}^{n_{max}} \sum_{\lambda_2=0}^{n_{max}} P_{\lambda_1, \lambda_2} z_1^{\lambda_1} z_2^{\lambda_2} \]  
(50)

with, \( n_{max}^P := max(2\lambda) \). The numerator coefficients \( P_{\lambda_1, \lambda_2} \) are matrices with dimensions \((p \times m)\).

\[ d(z_1, z_2) = \sum_{\lambda_1=0}^{n_{max}^q} \sum_{\lambda_2=0}^{n_{max}^q} q_{\lambda_1, \lambda_2} z_1^{\lambda_1} z_2^{\lambda_2} \]  
(51)

where, \( n_{max}^q := max(2\lambda) \). The denominator coefficients \( q_{\lambda_1, \lambda_2} \) are scalars.

The numerator polynomial matrix \( N(z_1, z_2) \) and the denominator polynomial \( d(z_1, z_2) \) can be numerically computed at \( R = (r + 1)^2 \), points, equally spaced on the unit 2D disc. The \( R \) points are chosen as \((z_1, z_2) = [v(i_1), v(i_2)], \ i_1, i_2 = 0, \cdots, r, \) with \( r = 2\lambda \), according to definition as:

\[ v_1(i) = v_2(i) = W^{-i}, \ \forall \ i = 0, \ldots, r. \]  
(52)

where,

\[ W_i = e^{(2\pi i) / (r+1)}, \ i = 1, 2 \]  
(53)

The values of the transfer function (47) at the \( R \) points are the corresponding 2D DFT coefficients.

4.2.1 Denominator Polynomial

To evaluate the denominator coefficients \( q_{\lambda_1, \lambda_2} \), define,

\[ a_{i_1, i_2} = \det [Ev_1^2(i_1)v_2^2(i_2) - A_0v_1(i_1)v_2(i_2) - A_1v_1(i_1) - A_2v_2(i_2)] \]  
(54)

Therefore using equations (51) and (54), \( a_{i_1, i_2} \) can be defined as,

\[ a_{i_1, i_2} = d[v_1(i_1), v_2(i_2)] \]  
(55)

Provided that at least one of \( a_{i_1, i_2} \neq 0 \).

Equations (51), (52) and (55) yield
\[ a_{i_1,i_2} = \sum_{\lambda_1=0}^{r} \sum_{\lambda_2=0}^{r} q_{\lambda_1,\lambda_2} W^{-(i_1\lambda_1+i_2\lambda_2)} \]  \hspace{1cm} (56)

In the above equation (56) it is obvious that \([a_{i_1,i_2}]\) and \([q_{\lambda_1,\lambda_2}]\) form a 2D DFT pair. Therefore the coefficients \([q_{\lambda_1,\lambda_2}]\) can be computed using the inverse 2D DFT, as follows:

\[ q_{\lambda_1,\lambda_2} = \frac{1}{R} \sum_{i_1=0}^{r} \sum_{i_2=0}^{r} a_{i_1,i_2} W^{(i_1\lambda_1+i_2\lambda_2)} \]  \hspace{1cm} (57)

\[ 4.2.2 \text{ Numerator Polynomial} \]

To evaluate the numerator matrix polynomial \(P_{\lambda_1,\lambda_2}\), define

\[ F_{i_1,i_2} = C \text{ adj} \left[ E v_1^2(i_1) v_2^2(i_2) - A_0 v_1(i_1) v_2(i_2) - A_1 v_1(i_1) - A_2 v_2(i_2) \right] \times [B_1 v_1(i_1) + B_2 v_2(i_2)] \]  \hspace{1cm} (58)

Using equations (50) and (58), \(F_{i_1,i_2}\) can be defined as,

\[ F_{i_1,i_2} = N[v_1(i_1), v_2(i_2)] \]  \hspace{1cm} (59)

Equations (50), (52) and (59) yield

\[ F_{i_1,i_2} = \sum_{\lambda_1=0}^{r-1} \sum_{\lambda_2=0}^{r-1} P_{\lambda_1,\lambda_2} W^{-(i_1\lambda_1+i_2\lambda_2)} \]  \hspace{1cm} (60)

In the above equation (60), \([F_{i_1,i_2}], [P_{\lambda_1,\lambda_2}]\) form a 2D DFT pair. Therefore the coefficients \(P_{\lambda_1,\lambda_2}\) can be computed, using the inverse 2D DFT, as follows:

\[ P_{\lambda_1,\lambda_2} = \frac{1}{R} \sum_{i_1=0}^{r-1} \sum_{i_2=0}^{r-1} F_{i_1,i_2} W^{(i_1\lambda_1+i_2\lambda_2)} \]  \hspace{1cm} (61)

Finally, the transfer function sought is,

\[ T(z_1, z_2) = \frac{N(z_1, z_2)}{d(z_1, z_2)} \]  \hspace{1cm} (62)

where,
\[ N(z_1, z_2) = \sum_{\lambda_1=0}^{r-1} \sum_{\lambda_2=0}^{r-1} P_{\lambda_1, \lambda_2} z_1^{\lambda_1} z_2^{\lambda_2} \]  

(63)

\[ d(z_1, z_2) = \sum_{\lambda_1=0}^{r} \sum_{\lambda_2=0}^{r} q_{\lambda_1, \lambda_2} z_1^{\lambda_1} z_2^{\lambda_2} \]  

(64)

A salient example, simple yet illustrative of the theoretical concepts presented in this work, follow below:

### 4.3 Example: G2O2D, two-inputs two-outputs, system

Consider the following generalized second–order 2D system with two–inputs and two–outputs:

\[
\begin{align*}
E x(i_1+2, i_2+2) &= A_0 x(i_1+1, i_2+1) + A_1 x(i_1+1, i_2) + A_2 x(i_1, i_2+1) \\
&\quad + B_1 u(i_1+1, i_2) + B_2 u(i_1, i_2+1) \\
y(i_1, i_2) &= C x(i_1, i_2)
\end{align*}
\]  

(65)

where,

\[
E = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad A_0 = \begin{bmatrix} 1 & 3 \\ 0 & 0 \end{bmatrix}, \quad A_1 = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}
\]

\[
B_1 = \begin{bmatrix} 1 & 2 \\ -3 & 0 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 3 & 1 \\ 0 & 1 \end{bmatrix}
\]

\[
C = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix}
\]

Since \( \lambda = 2 \), the \( r = 2 \times \lambda = 4 \). Therefore \( R = (r + 1)^2 = 25 \). The direct application of the proposed algorithm yields:
\[
\begin{bmatrix}
  a_{00} & a_{01} & a_{02} & a_{03} & a_{04} \\
  a_{10} & a_{11} & a_{12} & a_{13} & a_{14} \\
  a_{20} & a_{21} & a_{22} & a_{23} & a_{24} \\
  a_{30} & a_{31} & a_{32} & a_{33} & a_{34} \\
  a_{40} & a_{41} & a_{42} & a_{43} & a_{44}
\end{bmatrix}
= \\
\begin{bmatrix}
  -7.000 + 0.000j & 2.736 + 2.714j & -1.736 - 2.265j & -1.736 - 2.265j & 2.736 - 2.714j \\
  -0. + 3.804j & 2.045 + 1.763j & 2.045 - 0.587j & 1.882 + 0.812j & -1.309 - 2.853j \\
  1.618 + 2.351j & 4.118 - 3.441j & -3.545 - 2.853j & -0.191 - 1.763j & -3.545 + 0.951j \\
  1.618 - 2.351j & -3.545 - 0.951j & -0.191 + 1.763j & -3.545 + 0.951j & 4.118 + 3.441j \\
  -0.618 - 3.804j & 1.309 + 2.8532j & 1.882 - 0.812j & 2.045 + 0.587j & 2.045 - 1.763j \\
\end{bmatrix}
\]

and

\[
\begin{bmatrix}
  1.000 & 4.000 \\
  -45.000 & 2.000
\end{bmatrix}
= \\
\begin{bmatrix}
  -1.763 - 3.804j & 0.118 - 2.265j \\
  -14.798 + 30.156j & -9.972 - 1.987j
\end{bmatrix}
= \\
\begin{bmatrix}
  -6.236 - 2.351j & -2.118 - 2.714j \\
  9.798 + 10.751j & -1.027 + 5.343j
\end{bmatrix}
= \\
\begin{bmatrix}
  -6.236 + 2.351j & -2.118 + 2.714j \\
  9.798 - 10.751j & -1.027 - 5.343j
\end{bmatrix}
= \\
\begin{bmatrix}
  -1.763 + 3.804j & 0.118 + 2.265j \\
\end{bmatrix}
= \\
\begin{bmatrix}
  5.736 - 2.714j & 0.809 - 2.489j \\
  16.444 + 25.882j & 4.663 - 2.040j
\end{bmatrix}
= \\
\begin{bmatrix}
  -6.236 + 1.175j & -1.427 - 2.938j \\
  30.978 + 7.053j & 0.191 + 5.290j
\end{bmatrix}
\]
\[
\begin{align*}
F_{12} &= \begin{bmatrix}
5.899 + 9.820j & -3.927 + 3.302j \\
16.281 - 16.531j & 8.545 + 3.302j
\end{bmatrix} \\
F_{13} &= \begin{bmatrix}
5.736 - 2.9919j & 2.354 + 1.538j \\
-11.972 - 5.898j & -0.500 - 3.441j
\end{bmatrix} \\
F_{14} &= \begin{bmatrix}
1.000 + 3.526j & 2.191 + 0.587j \\
\end{bmatrix} \\
F_{20} &= \begin{bmatrix}
1.263 + 2.265j & -0.309 - 0.224j \\
-1.444 - 14.233j & -3.163 - 5.204j
\end{bmatrix} \\
F_{21} &= \begin{bmatrix}
1.263 - 5.792j & -4.354 - 0.363j \\
-3.027 + 18.102j & -0.500 - 0.812j
\end{bmatrix} \\
F_{22} &= \begin{bmatrix}
-1.763 - 1.902j & 1.927 + 4.755j \\
\end{bmatrix} \\
F_{23} &= \begin{bmatrix}
1.000 - 5.706j & 3.309 - 0.951j \\
-2.763 + 2.971j & -6.781 - 1.228j
\end{bmatrix} \\
F_{24} &= \begin{bmatrix}
-6.399 - 3.130j & -0.572 - 3.216j \\
6.218 - 11.531j & 2.954 - 3.216j
\end{bmatrix} \\
F_{30} &= \begin{bmatrix}
1.263 - 2.265j & -0.309 + 0.224j \\
-1.444 + 14.233j & -3.163 + 5.204j
\end{bmatrix} \\
F_{31} &= \begin{bmatrix}
-6.399 + 3.130j & -0.572 + 3.216j \\
6.218 + 11.531j & 2.954 + 3.216j
\end{bmatrix} \\
F_{32} &= \begin{bmatrix}
1.000 + 5.706j & 3.309 + 0.951j \\
-2.763 - 2.971j & -6.781 + 1.228j
\end{bmatrix} \\
F_{33} &= \begin{bmatrix}
-1.763 + 1.902j & 1.927 - 4.755j \\
\end{bmatrix} \\
F_{34} &= \begin{bmatrix}
1.263 + 5.792j & -4.354 + 0.363j \\
-3.027 + 18.102j & -0.500 + 0.812j
\end{bmatrix}
\end{align*}
\]
\[
F_{40} = \begin{bmatrix}
5.736 + 2.714j & 0.809 + 2.489j \\
16.444 - 25.882j & 4.663 + 2.040j
\end{bmatrix}
\]

\[
F_{41} = \begin{bmatrix}
1.000 - 3.526j & 2.191 - 0.587j \\
\end{bmatrix}
\]

\[
F_{42} = \begin{bmatrix}
5.736 + 2.991j & 2.354 - 1.538j \\
-11.972 + 5.898j & -0.500 + 3.441j
\end{bmatrix}
\]

\[
F_{43} = \begin{bmatrix}
5.899 - 9.820j & -3.927 - 3.302j \\
16.281 + 16.531j & 8.545 - 3.302j
\end{bmatrix}
\]

\[
F_{44} = \begin{bmatrix}
-6.236 - 1.175j & -1.427 + 2.938j \\
30.978 - 7.053j & 0.191 - 5.290j
\end{bmatrix}
\]

Using (57), the denominator coefficients are,

\[
\begin{bmatrix}
q_{00} & q_{01} & q_{02} & q_{03} & q_{04} \\
q_{10} & q_{11} & q_{12} & q_{13} & q_{14} \\
q_{20} & q_{21} & q_{22} & q_{23} & q_{24} \\
q_{30} & q_{31} & q_{32} & q_{33} & q_{34} \\
q_{40} & q_{41} & q_{42} & q_{43} & q_{44}
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & -1 & 0 & 0 \\
0 & -2 & -2 & 0 & 0 \\
-1 & 1 & 0 & -1 & 0 \\
0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

Using (61), the numerator matrix polynomials are,

\[
\begin{bmatrix}
P_{00} & P_{01} & P_{02} & P_{03} & P_{04} \\
P_{10} & P_{11} & P_{12} & P_{13} & P_{14} \\
P_{20} & P_{21} & P_{22} & P_{23} & P_{24} \\
P_{30} & P_{31} & P_{32} & P_{33} & P_{34} \\
P_{40} & P_{41} & P_{42} & P_{43} & P_{44}
\end{bmatrix}
\]

where the matrices \( P_{00}, \ldots, P_{44} \) are,
or

\[ T(z_1, z_2) = \frac{P_{02}z_2^2 + P_{11}z_1z_2 + P_{12}z_1z_2^2 + P_{20}z_1^2 + P_{21}z_1^2z_2 + P_{23}z_1^2z_2^2 + P_{32}z_1^3z_2^2}{q_{02}z_2 + q_{11}z_1z_2 + q_{12}z_1z_2^2 + q_{20}z_1^2 + q_{21}z_1^2z_2 + q_{23}z_1^2z_2^2 + q_{32}z_1^3z_2^2} \]

or

\[ T(z_1, z_2) = \begin{bmatrix} 0 & 0 & \begin{bmatrix} 3 & 1 \\ -3 & 1 \end{bmatrix} z_2^2 & 0 & 0 \\ 0 & \begin{bmatrix} 1 & 3 \\ -13 & -1 \end{bmatrix} z_1z_2 & 0 & -1 \\ 0 & \begin{bmatrix} 3 & 0 \\ -15 & 0 \end{bmatrix} z_1^2z_2 & 0 & 0 \\ 0 & \begin{bmatrix} -3 & 0 \\ -3 & 0 \end{bmatrix} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1z_2^2 \\ z_1^2z_2 \\ z_1^2z_2^3 \\ z_1^3z_2^2 \\ -z_2^2 - 2z_1z_2 - 2z_1z_2^2 - z_1^2z_2 - z_1^2z_2^3 - z_1^3z_2^2 \end{bmatrix} \]

Once the denominator and the adjoint matrix coefficients have been computed, equation (62) can be utilized to obtain the transfer function \( T(z_1, z_2) \). Therefore, we have
or

\[
T(z_1, z_2) = \begin{bmatrix}
\alpha_{11} & \alpha_{12} \\
\alpha_{21} & \alpha_{22}
\end{bmatrix}
\]

\[
= \frac{\alpha_{11}}{-z_2^2 - 2z_1z_2 - 2z_1z_2^2 - z_1^2 + z_1^2z_2 - z_1^2z_2^2 - z_1^2z_2^2}
\]

(66)

where,

\[
\alpha_{11} = 3z_2^2 + z_1z_2 - 3z_1^2 + 3z_1^2z_2 - 3z_1z_2^2
\]

\[
\alpha_{12} = z_2^2 + 3z_1z_2 - z_1^2z_2 + z_1^2z_2
\]

\[
\alpha_{21} = -3z_2^2 - 13z_1z_2 - 11z_1^2 - 15z_1^2z_2 - 3z_1z_2^2
\]

\[
\alpha_{22} = z_2^2 - z_1z_2 + 5z_1z_2^2 - 4z_1^2 + z_1^2z_2
\]

The correctness of the above result (66) can easily be verified using (42).

### 4.4 Complexity of the Algorithm

The proposed algorithm has two parts. In the first part the matrices \( F_{i_1,i_2} \) and the scalars \( a_{i_1,i_2} \) are evaluated with a cost of \( pmR\lambda^3 \) operations. In the second part the coefficients of \( P_{\lambda_1,\lambda_2} \) and \( q_{\lambda_1,\lambda_2} \) are evaluated using the DFT with a cost of \( pmR^2 + R^2 \) operations. For more efficient computation, especially for high order systems, fast Fourier methods can be used to implement the DFT [31].

Due to the inherent modularity and the algorithmic structure of the presented method high parallelism is permitted. In this case the computation of each determinant \( a_{i_1,i_2} \), (56), and each matrix product \( F_{i_1,i_2} \), (60), can be distributed over a number of processing elements, considerably reducing the computation time of the algorithm.
A multi–dimensional (nD) and multi–order k–Order, (nDkO), system is described by the following set of equations:

\[
x(i_1 + k, i_2 + k, i_3 + k, \ldots, i_n + k) = \sum_{\lambda=1}^{k-1} A_0^\lambda x(i_1 + \lambda, i_2 + \lambda, i_3 + \lambda, \ldots, i_n + \lambda) \\
+ \sum_{\mu=1}^n A_\mu x(i_1 + \nu_1, i_2 + \nu_2, i_3 + \nu_3, \ldots, i_n + \nu_n) \\
+ \sum_{\mu=1}^n B_\mu u(i_1 + \nu_1, i_2 + \nu_2, i_3 + \nu_3, \ldots, i_n + \nu_n)
\]

where, \(\nu_j = \begin{cases} 1 & \text{for } \mu = j \\ 0 & \text{for } \mu \neq j \end{cases}\)

\[
y(i_1, i_2, i_3, \ldots, i_n) = C x(i_1, i_2, i_3, \ldots, i_n)
\]

It is noted that the nDkO system (67) is an extension of the 2D model ([4],[9]) to nD and k-Order.

Applying the nD \(z_i\), \((i = 1, 2, \ldots, n)\) transform to the system (67), with zero initial conditions, the corresponding transfer function becomes:

\[
T(z_1, z_2, \ldots, z_n) = C [I_{z_1^k} z_2^k \cdots z_n^k - \sum_{\lambda=1}^{k-1} A_0^\lambda z_1^\lambda z_2^\lambda \cdots z_n^\lambda - \sum_{\mu=1}^n A_\mu z_\mu]^{-1} \cdot [\sum_{\mu=1}^n B_\mu z_\mu]
\]

In the following section an interpolative approach is developed for determining the transfer function \(T(z_1, z_2, \ldots, z_n)\), given the matrices, \(A_\lambda^0\) for \(\lambda = 1, \ldots, k-1\), \(A_\mu, B_\mu\) for
\( \mu = 1, \ldots, n \) and \( C \), using the DFT. For the sake of completeness a brief description of the DFT follows.

### 5.1 nD DFT

Consider the finite sequences \( X(k_1, k_2, \ldots, k_n) \) and \( \tilde{X}(r_1, r_2, \ldots, r_n) \), \( k_i, r_i = 0, \ldots, M_i \), \( \forall i = 1, 2, \ldots, n \). In order for the sequences \( X(k_1, k_2, \ldots, k_n) \) and \( \tilde{X}(r_1, r_2, \ldots, r_n) \), to constitute a nD DFT pair the following relations should hold [10]:

\[
\tilde{X}(r_1, r_2, \ldots, r_n) = M_1 \sum_{k_1=0}^{M_1} \cdots \sum_{k_n=0}^{M_n} X(k_1, k_2, \ldots, k_n) \times W_1^{-k_1 r_1} W_2^{-k_2 r_2} \cdots W_n^{-k_n r_n} \\
X(k_1, k_2, \ldots, r_n) = \frac{1}{R} \sum_{r_1=0}^{M_1} \cdots \sum_{r_n=0}^{M_n} \tilde{X}(r_1, r_2, \ldots, r_n) \times W_1^{k_1 r_1} W_2^{k_2 r_2} \cdots W_n^{k_n r_n}
\]

(69)

(70)

where,

\[
R = \prod_{i=1}^{n} (M_i + 1)
\]

(71)

\[
W_i = e^{(2\pi j)/(M_i+1)}, \quad \forall i = 1, 2, \ldots, n
\]

(72)

\( X, \tilde{X} \) are discrete argument matrix valued functions, with dimensions \( p \times m \).

In the following sections an interpolative approach is developed for determining the transfer function \( T(z_1, z_2, \ldots, z_n) \), given the matrices \( A_0^\lambda \) for \( \lambda = 1, \ldots, k-1 \), \( A_\mu, B_\mu \) for \( \mu = 1, \ldots, n \) and \( C \), using the DFT.

### 5.2 Algorithm

The transfer function of the \( nDkO \) system (67) has the structure,

\[
T(z_1, z_2, \ldots, z_n) = \frac{N(z_1, z_2, \ldots, z_n)}{d(z_1, z_2, \ldots, z_n)}
\]

(73)

where,
\[
N(z_1, z_2, \ldots, z_n) = C \text{adj} \left[ I z_1^k z_2^k \cdots z_n^k - \sum_{\lambda=1}^{k-1} A_0^\lambda z_1^\lambda z_2^\lambda \cdots z_n^\lambda - \sum_{\mu=1}^{n} A_\mu z_\mu \right] \\
\times \sum_{\mu=1}^{n} B_\mu z_\mu 
\] (74)

\[
d(z_1, z_2, \ldots, z_n) = \det \left[ I z_1^k z_2^k \cdots z_n^k - \sum_{\lambda=1}^{k-1} A_0^\lambda z_1^\lambda z_2^\lambda \cdots z_n^\lambda - \sum_{\mu=1}^{n} A_\mu z_\mu \right] 
\] (75)

Equations (74) and (75) can be written in polynomial form as follows:

\[
N(z_1, z_2, \ldots, z_n) = \sum_{\lambda_1=0}^{n (P)} \sum_{\lambda_2=0}^{n (P)} \cdots \sum_{\lambda_n=0}^{n (P)} P_{\lambda_1, \lambda_2, \ldots, \lambda_n} \cdot z_1^{\lambda_1} z_2^{\lambda_2} \cdots z_n^{\lambda_n} 
\] (76)

with, \( n_{\text{max}}(P) := \max(n-1)(k-1), (n-1) \). The numerator coefficient matrices, \( P_{\lambda_1, \lambda_2, \ldots, \lambda_n} \) have dimensions \((p \times m)\).

\[
d(z_1, z_2, \ldots, z_n) = \sum_{\lambda_1=0}^{n (q)} \sum_{\lambda_2=0}^{n (q)} \cdots \sum_{\lambda_n=0}^{n (q)} q_{\lambda_1, \lambda_2, \ldots, \lambda_n} \cdot z_1^{\lambda_1} z_2^{\lambda_2} \cdots z_n^{\lambda_n} 
\] (77)

with, \( n_{\text{max}}(q) := \max(n(k-1), n) \). The denominator coefficients \( q_{\lambda_1, \lambda_2, \ldots, \lambda_n} \) are scalars.

The numerator polynomial matrix \( N(z_1, z_2, \ldots, z_n) \) and the denominator polynomial \( d(z_1, z_2, \ldots, z_n) \) can be numerically computed at \( R = \prod_{i=1}^{n} (r+1) = (r+1)^n \) points equally spaced on the unit \( nD \) hypersphere, with \( r = n \times \gamma \). The \( R \) points are chosen as

\[
v_1(i) = v_2(i) = \cdots = v_n(i) = W^{-i}, \ \forall \ i = 0, \ldots, r 
\] (78)

where,

\[
W_i = W = e^{(2\pi j)/(r+1)}, \ \forall \ i = 1, 2, \ldots, r 
\] (79)

The values of the transfer function at the \( R \) points are the corresponding \( nD \) DFT coefficients.
5.2.1 Denominator Polynomial

To evaluate the denominator coefficients \( (q_{\lambda_1, \lambda_2, \ldots, \lambda_n}) \), define,

\[
a_{i_1, i_2, \ldots, i_n} = \det \left[ I_{v_1^k(i_1)} v_2^k(i_2) \cdots v_n^k(i_n) - \sum_{\lambda=1}^{k-1} A_\lambda^\lambda v_1^\lambda(i_1) v_2^\lambda(i_2) \times v_n^\lambda(i_n) - \sum_{\mu=1}^n A_\mu v_\mu(i_\mu) \right] \quad (80)
\]

Therefore using equations (77) and (80), yields

\[
a_{i_1, i_2, \ldots, i_n} = d[v_1(i_1), v_2(i_2), \ldots, v_n(i_n)] \quad (81)
\]

Provided that at least one of \( a_{i_1, i_2, \ldots, i_n} \neq 0 \).

Equations (77), (78) and (81) yield

\[
a_{i_1, i_2, \ldots, i_n} = \sum_{\lambda_1=0}^{r} \sum_{\lambda_2=0}^{r} \cdots \sum_{\lambda_n=0}^{r} q_{\lambda_1, \lambda_2, \ldots, \lambda_n} \cdot W^{-1}(i_1 \lambda_1 + i_2 \lambda_2 + \cdots + i_n \lambda_n) \quad (82)
\]

In the above equation (82) \([a_{i_1, i_2, \ldots, i_n}]\) and \([q_{\lambda_1, \lambda_2, \ldots, \lambda_n}]\) form a DFT pair. Therefore the coefficients \([q_{\lambda_1, \lambda_2, \ldots, \lambda_n}]\) can be computed using the inverse \( nD \) DFT, as follows:

\[
q_{\lambda_1, \lambda_2, \ldots, \lambda_n} = \frac{1}{R} \sum_{i_1=0}^{r} \sum_{i_2=0}^{r} \cdots \sum_{i_n=0}^{r} a_{i_1, i_2, \ldots, i_n} \cdot W^{1}(i_1 \lambda_1 + i_2 \lambda_2 + \cdots + i_n \lambda_n) \quad (83)
\]

5.2.2 Numerator Polynomial

To evaluate the numerator matrix polynomial \( P_{\lambda_1, \lambda_2, \ldots, \lambda_n} \), define,

\[
F_{i_1, i_2, \ldots, i_n} = C \text{adj} \left[ I_{v_1^k(i_1)} v_2^k(i_2) \cdots v_n^k(i_n) - \sum_{\lambda=1}^{k-1} A_\lambda^\lambda v_1^\lambda(i_1) v_2^\lambda(i_2) \times v_n^\lambda(i_n) - \sum_{\mu=1}^n A_\mu v_\mu(i_\mu) \right] \quad (84)
\]

Therefore using equations (76) and (84), yields

\[
F_{i_1, i_2, \ldots, i_n} = N[v_1(i_1), v_2(i_2), \ldots, v_n(i_n)] \quad (85)
\]

Equations (76), (78) and (85) yield
\[ F_{i_1, i_2, \ldots, i_n} = \sum_{\lambda_1=0}^{r-1} \sum_{\lambda_2=0}^{r-1} \cdots \sum_{\lambda_n=0}^{r-1} P_{\lambda_1, \lambda_2, \ldots, \lambda_n} \cdot W^{-(i_1 \lambda_1 + i_2 \lambda_2 + \cdots + i_n \lambda_n)} \] (86)

In the above equation (86) it is obvious that \([F_{i_1, i_2, \ldots, i_n}]\) and \([P_{\lambda_1, \lambda_2, \ldots, \lambda_n}]\) form a \(nD\) DFT pair. Therefore the coefficients \(P_{\lambda_1, \lambda_2, \ldots, \lambda_n}\) can be computed, using the inverse \(nD\) DFT, as follows:

\[ P_{\lambda_1, \lambda_2, \ldots, \lambda_n} = \frac{1}{R} \sum_{i_1=0}^{r} \sum_{i_2=0}^{r} \cdots \sum_{i_n=0}^{r} F_{i_1, i_2, \ldots, i_n} \cdot W^{(i_1 \lambda_1 + i_2 \lambda_2 + \cdots + i_n \lambda_n)} \] (87)

Finally, the transfer function sought is,

\[ T(z_1, z_2, \ldots, z_n) = \frac{N(z_1, z_2, \ldots, z_n)}{d(z_1, z_2, \ldots, z_n)} \] (88)

where,

\[ N(z_1, z_2, \ldots, z_n) = \sum_{\lambda_1=0}^{r-1} \sum_{\lambda_2=0}^{r-1} \cdots \sum_{\lambda_n=0}^{r-1} P_{\lambda_1, \lambda_2, \ldots, \lambda_n} \cdot z_1^{\lambda_1} z_2^{\lambda_2} \cdots z_n^{\lambda_n} \] (89)

\[ d(z_1, z_2, \ldots, z_n) = \sum_{\lambda_1=0}^{r-1} \sum_{\lambda_2=0}^{r-1} \cdots \sum_{\lambda_n=0}^{r-1} q_{\lambda_1, \lambda_2, \ldots, \lambda_n} \cdot z_1^{\lambda_1} z_2^{\lambda_2} \cdots z_n^{\lambda_n} \] (90)

In the next section three illustrative examples are given. The mathematical software package Matlab\textsuperscript{TM} was used for the implementation of the algorithm.

### 5.3 Complexity of the Algorithm

The proposed algorithm has two parts –In the first part the matrices \(F_{i_1, i_2, \ldots, i_n}\) and the scalars \(a_{i_1, i_2, \ldots, i_n}\) are evaluated with a cost of \(pmRn^3\) operations. In the second part the coefficients of \(P_{\lambda_1, \lambda_2, \ldots, \lambda_n}\) and \(q_{\lambda_1, \lambda_2, \ldots, \lambda_n}\) are evaluated using the DFT with a cost of \(pmR^2 + R^2\) operations. For more efficient computation, especially for high order systems, fast Fourier methods can be used to implement the DFT [31].

Due to the inherent modularity and the algorithmic structure of the presented method high parallelism is permitted. In this case the computation of each determinant \(a_i\), (82), and each matrix product \(F_i\), (86), can be distributed over a number of processing elements, considerably reducing the computation time of the algorithm.
A Generalized (singular) $k$-Order $n$-Dimensional ($SkOnD$) model is described by the following set of equations:

$$
\mathbf{E}x(i_1 + k, i_2 + k, i_3 + k, \ldots, i_n + k) = + \sum_{\lambda=1}^{k-1} \mathbf{A}_\lambda x(i_1 + \lambda, i_2 + \lambda, i_3 + \lambda, \ldots, i_n + \lambda)
$$

$$
+ \sum_{\mu=1}^{n} \mathbf{A}_\mu x(i_1 + \nu_1, i_2 + \nu_2, i_3 + \nu_3, \ldots, i_n + \nu_n)
$$

$$
\nu_j = \begin{cases} 
1 & \text{for } \mu = j \\
0 & \text{for } \mu \neq j 
\end{cases}
$$

$$
+ \sum_{\mu=1}^{n} \mathbf{B}_\mu u(i_1 + \nu_1, i_2 + \nu_2, i_3 + \nu_3, \ldots, i_n + \nu_n)
$$

$$
\nu_j = \begin{cases} 
1 & \text{for } \mu = j \\
0 & \text{for } \mu \neq j 
\end{cases}
$$

(91)

$$
y(i_1, i_2, i_3, \ldots, i_n) = Cx(i_1, i_2, i_3, \ldots, i_n)
$$

where, $x(i_1, i_2, i_3, \ldots, i_n) \in \mathbb{R}^n$, $u(i_1, i_2, i_3, \ldots, i_n) \in \mathbb{R}^m$, $y(i_1, i_2, i_3, \ldots, i_n) \in \mathbb{R}^p$; $\mathbf{A}_k$, $k = 1, \ldots, n$, $\mathbf{E}$, $\mathbf{B}$, $\mathbf{C}$, are real matrices of appropriate dimensions. matrix $\mathbf{E}$ may be singular with rank $\epsilon$.

It is noted that the $SkOnD$ state space model (91) is an extension of the 2D Fornasini–Marchesini model [3] to $n$-Dimensions and $k$-Order. For more 2D second-order model structures the reader can refer to [10].

Applying the $nD$ $z_i$, ($i = 1, 2, 3, \ldots n$) transform to the system (91), with zero initial conditions, the transfer function of (91) becomes:

$$
T_1(z_1, z_2, z_3, \ldots, z_n) = C [\mathbf{E} z_1^k z_2^k z_3^k \cdots z_n^k - \sum_{\lambda=1}^{k-1} \mathbf{A}_\lambda z_1^\lambda z_2^\lambda \cdots z_n^\lambda - \sum_{\mu=1}^{n} \mathbf{A}_\mu z_\mu]^{-1}
$$
\[
\sum_{\mu=1}^{n} B_{\mu} z_{\mu}
\]

(92)

In the following section an interpolative approach is developed for determining the transfer function \(T(z_1, z_2, z_3, \ldots, z_n)\), given the matrices \(A_k\) for \(k = 1, \ldots, n\) and \(E, B, C\), using the DFT. For the sake of completeness a brief description of the DFT follows.

6.1 \(nD\ DFT\)

Consider the finite sequences \(X(k_1, k_2, k_3, \ldots, k_n)\) and \(\tilde{X}(r_1, r_2, r_3, \ldots, r_n)\), \(k_i, r_i = 0, \ldots, M_i\ \forall \ i = 1, 2, 3, \ldots, n\). In order for the sequences \(X(k_1, k_2, k_3, \ldots, k_n)\) and \(\tilde{X}(r_1, r_2, r_3, \ldots, r_n)\), to constitute a \(nD\) DFT pair the following relations should hold [10]:

\[
\tilde{X}(r_1, r_2, r_3, \ldots, r_n) = \sum_{k_1=0}^{M_1} \sum_{k_2=0}^{M_2} \cdots \sum_{k_n=0}^{M_n} X(k_1, k_2, k_3, \ldots, k_n)
\times W_{-k_1r_1} W_{-k_2r_2} W_{-k_3r_3} \cdots W_{-k_nr_n}
\]

(93)

\[
X(k_1, k_2, k_3, \ldots, r_n) = \frac{1}{R} \sum_{r_1=0}^{M_1} \sum_{r_2=0}^{M_2} \cdots \sum_{r_n=0}^{M_n} \tilde{X}(r_1, r_2, r_3, \ldots, r_n)
\times W_{k_1r_1} W_{k_2r_2} W_{k_3r_3} \cdots W_{k_nr_n}
\]

(94)

where,

\[
R = \prod_{i=1}^{n} (M_i + 1)
\]

(95)

\[
W_i = e^{(2\pi j)/(M_i+1)}, \ \forall \ i = 1, 2, 3, \ldots, n
\]

(96)

\(X, \tilde{X}\) are discrete argument matrix valued functions, with dimensions \(p \times m\).

In the following section an interpolative approach is developed for determining the transfer function \(T(z_1, z_2, z_3, \ldots, z_n)\), given the matrices \(A_i, i = 1, \ldots, n, E, B\) and \(C\), using the DFT.

6.2 Algorithm

The transfer function of the \(SkOnD\) state space model (91) has the structure,
\[
\mathbf{T}(z_1, z_2, z_3, \ldots, z_n) = \frac{N(z_1, z_2, z_3, \ldots, z_n)}{d(z_1, z_2, z_3, \ldots, z_n)}
\] (97)

where,

\[
N(z_1, z_2, z_3, \ldots, z_n) = C \text{adj} \left[ E z_1^k z_2^k z_3^k \cdots z_n^k - \sum_{\lambda=1}^{k-1} A_0^\lambda z_1^\lambda z_2^\lambda \cdots z_n^\lambda - \sum_{\mu=1}^{n} A_\mu z_\mu \right]
\times \sum_{\mu=1}^{n} B_\mu z_\mu
\] (98)

\[
d(z_1, z_2, z_3, \ldots, z_n) = \text{det} \left[ I z_1^k z_2^k z_3^k \cdots z_n^k - \sum_{\lambda=1}^{k-1} A_0^\lambda z_1^\lambda z_2^\lambda \cdots z_n^\lambda - \sum_{\mu=1}^{n} A_\mu z_\mu \right]
\] (99)

It is noted that the (partial) degrees of the denominator are equal to the degree of the system.

Equations (98) and (99) can be written in polynomial form as follows:

\[
N(z_1, z_2, z_3, \ldots, z_n) = \sum_{\lambda_1=0}^{n-1} \sum_{\lambda_2=0}^{n-1} \sum_{\lambda_3=0}^{n-1} \cdots \sum_{\lambda_n=0}^{n-1} P_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n} z_1^{\lambda_1} z_2^{\lambda_2} z_3^{\lambda_3} \cdots z_n^{\lambda_n}
\] (100)

\[
d(z_1, z_2, z_3, \ldots, z_n) = \sum_{\lambda_1=0}^{n} \sum_{\lambda_2=0}^{n} \sum_{\lambda_3=0}^{n} \cdots \sum_{\lambda_n=0}^{n} q_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n} z_1^{\lambda_1} z_2^{\lambda_2} z_3^{\lambda_3} \cdots z_n^{\lambda_n}
\] (101)

\[P_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n}\] are matrices with dimensions \((p \times m)\), while \(q_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n}\) are scalars.

The numerator polynomial matrix \(N(z_1, z_2, z_3, \ldots, z_n)\) and the denominator polynomial \(d(z_1, z_2, z_3, \ldots, z_n)\) can be numerically computed at \(R = n \prod_{i=1}^{n}(r+1) = (r+1)^n\) points equally spaced on the unit \(nD\) hypersphere, with \(r = n \times \gamma\). The \(R\) points are chosen as

\[
v_1(i) = v_2(i) = v_3(i) = \cdots = v_n(i) = W^{-i}, \ \forall \ i = 0, \ldots, r
\] (102)

where,

\[
W_i = W = e^{(2\pi j)/(r+1)}, \ \forall \ i = 1, 2, 3, \ldots, r
\] (103)

The values of the transfer function (97) at the \(R\) points are the corresponding \(nD\) DFT coefficients.
6.2.1 Denominator Polynomial

To evaluate the denominator coefficients \( q_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n} \), define,

\[
a_{i_1, i_2, i_3, \ldots, i_n} = \text{det} \left[ \mathbf{E}v_k^1(i_1)v_k^2(i_2)v_k^3(i_3) \cdots v_k^n(i_n) \right. \\
- \sum_{\lambda_1=1}^{k-1} A_{\lambda_1}^1v_{\lambda_1}^1(i_1)v_{\lambda_2}^2(i_2)v_{\lambda_3}^3(i_3) \cdots v_{\lambda_n}^n(i_n) \\
- \sum_{\mu=1}^{n} A_{\mu}v_{\mu}(i_\mu) \right] 
\]

Therefore using equations (101) and (104) yields

\[
a_{i_1, i_2, i_3, \ldots, i_n} = d[v_1(i_1), v_2(i_2), v_3(i_3), \ldots, v_n(i_n)] 
\]

Provided that at least one of \( a_{i_1, i_2, i_3, \ldots, i_n} \neq 0 \).

Equations (101), (102) and (105) yield

\[
a_{i_1, i_2, i_3, \ldots, i_n} = \sum_{\lambda_1=0}^{r} \sum_{\lambda_2=0}^{r} \sum_{\lambda_3=0}^{r} \cdots \sum_{\lambda_n=0}^{r} q_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n} W^{-(i_1\lambda_1+i_2\lambda_2+i_3\lambda_3+\ldots+i_n\lambda_n)} 
\]

In the above equation (106) \([a_{i_1, i_2, i_3, \ldots, i_n}]\) and \([q_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n}]\) form a DFT pair. Therefore the coefficients \([q_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n}]\) can be computed using the inverse \( nD \) DFT, as follows:

\[
q_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n} = \frac{1}{R} \sum_{i_1=0}^{r} \sum_{i_2=0}^{r} \sum_{i_3=0}^{r} \cdots \sum_{i_n=0}^{r} a_{i_1, i_2, i_3, \ldots, i_n} W^{(i_1\lambda_1+i_2\lambda_2+i_3\lambda_3+\ldots+i_n\lambda_n)} 
\]

6.2.2 Numerator Polynomial

To evaluate the numerator matrix polynomial \( P_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n} \), define,

\[
F_{i_1, i_2, i_3, \ldots, i_n} = \text{adj} \left[ \mathbf{E}v_k^1(i_1)v_k^2(i_2)v_k^3(i_3) \cdots v_k^n(i_n) \right. \\
- \sum_{\lambda_1=1}^{k-1} A_{\lambda_1}^1v_{\lambda_1}^1(i_1)v_{\lambda_2}^2(i_2)v_{\lambda_3}^3(i_3) \cdots v_{\lambda_n}^n(i_n) \\
- \sum_{\mu=1}^{n} A_{\mu}v_{\mu}(i_\mu) \right] \times \left[ \sum_{\mu=1}^{n} B_{\mu}v_{\mu}(i_\mu) \right]
\]
Therefore using equations (100) and (108), yields

\[ F_{i_1, i_2, i_3, \ldots, i_n} = N[v_1(i_1), v_2(i_2), v_3(i_3), \ldots, v_n(i_n)] \quad (109) \]

Equations (100), (102) and (109) yield

\[ F_{i_1, i_2, i_3, \ldots, i_n} = r - \sum_{\lambda_1=0}^{r-1} \lambda_1 = 0 \sum_{\lambda_2=0}^{r-1} \lambda_2 \sum_{\lambda_3=0}^{r-1} \lambda_3 \ldots \sum_{\lambda_n=0}^{r-1} \lambda_n W^{-i_1 \lambda_1 + i_2 \lambda_2 + i_3 \lambda_3 + \cdots + i_n \lambda_n} \quad (110) \]

In the above equation (110) it is obvious that \([F_{i_1, i_2, i_3, \ldots, i_n}]\) and \([P_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n}]\) form an \(nD\) DFT pair. Therefore the coefficients \(P_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n}\) can be computed, using the inverse \(nD\) DFT, as follows:

\[ P_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n} = \frac{1}{R} \sum_{i_1=0}^{r-1} \sum_{i_2=0}^{r-1} \sum_{i_3=0}^{r-1} \sum_{i_n=0}^{r-1} F_{i_1, i_2, i_3, \ldots, i_n} W^{i_1 \lambda_1 + i_2 \lambda_2 + i_3 \lambda_3 + \cdots + i_n \lambda_n} \quad (111) \]

Finally, the transfer function sought is,

\[ T(z_1, z_2, z_3, \ldots, z_n) = \frac{N(z_1, z_2, z_3, \ldots, z_n)}{d(z_1, z_2, z_3, \ldots, z_n)} \quad (112) \]

where,

\[ N(z_1, z_2, z_3, \ldots, z_n) = \sum_{\lambda_1=0}^{r-1} \sum_{\lambda_2=0}^{r-1} \sum_{\lambda_3=0}^{r-1} \sum_{\lambda_n=0}^{r-1} P_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n} z_1^{\lambda_1} z_2^{\lambda_2} z_3^{\lambda_3} \cdots z_n^{\lambda_n} \quad (113) \]

\[ d(z_1, z_2, z_3, \ldots, z_n) = \sum_{\lambda_1=0}^{r-1} \sum_{\lambda_2=0}^{r-1} \sum_{\lambda_3=0}^{r-1} \sum_{\lambda_n=0}^{r-1} q_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n} z_1^{\lambda_1} z_2^{\lambda_2} z_3^{\lambda_3} \cdots z_n^{\lambda_n} \quad (114) \]

### 6.3 Complexity of the Algorithm

The proposed algorithm has two parts. In the first part the matrices \(F_{i_1, i_2, \ldots, i_n}\) and the scalars \(a_{i_1, i_2, \ldots, i_n}\) are evaluated with a cost of \(pmRn^3\) operations. In the second part the coefficients of \(P_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n}\) and \(q_{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n}\) are evaluated using the DFT with a cost of \(pmR^2 + R^2\) operations. For more efficient computation, especially for high order systems, fast Fourier methods can be used to implement the DFT [31].
Due to the inherent modularity and the algorithmic structure of the presented method high parallelism is permitted. In this case the computation of each determinant \( a_i \), (104), and each matrix product \( F_i \), (108), can be distributed over a number of processing elements, considerably reducing the computation time of the algorithm.
CHAPTER 7

CONCLUSION

In this thesis the DFT was used for computing the transfer functions for the following systems/models:

- Second-order 2D system
- Generalized second-order 2D system
- \(k\)-order \(n\)-dimensional system
- Generalized \(k\)-order \(n\)-dimensional system

The algorithms were implemented with the software package \(Matlab^{TM}\). To further improve the computational speed of the algorithm, fast Fourier techniques and computer systems with multiple central processing units can be used.

Moreover, two second-order 2D models were extended to \(n\)-dimensional \(k\)-order (multi-dimensional multi-order) for regular and generalized systems.

As it can be seen from the 2D second-order examples the presented models can realize 2D transfer functions in a more compact form having lower matrix-dimensions, \((2 \times 2)\), than the classical 2D models that require \((4 \times 4)\) matrices. This is also the case for higher dimension-order systems.
CHAPTER 8

FUTURE WORK

Using the regular or generalized 2(k)–order 2(n)–D models the following problems can be considered.

• Stability,

• Solvability and properties,

• Reachability and observability,

• Model equivalence,

• Regularity,

• Geometric theory,

• Minimum energy control,

• More (faster) techniques for transfer function computation,

• State–space and circuit realization etc.,

• Positive k-order nD system–based problems

• VHDL/FPGA implementations

• Applications to filtering
MATLAB™ code for computing the transfer function coefficients using the 2D DFT.

\[
\begin{align*}
E &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\
A_0 &= \begin{bmatrix} 1 & 3 \\ 0 & 0 \end{bmatrix} \\
A_1 &= \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix} \\
A_2 &= \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \\
B_1 &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
B_2 &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
C &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
D &= 0
\end{align*}
\]

%%----------------------------------------------------------------------%%

\[
\begin{align*}
I &= \sqrt{-1} \\
N &= 2; \\
W_{\text{temp}} &= \exp(2\pi i/5); \\
\text{clear } Z \\
Z(1,1) &= 1; \\
Z(1,2) &= W_{\text{temp}}^{-1}; \\
Z(1,3) &= W_{\text{temp}}^{-2};
\end{align*}
\]
\[ Z(1,4) = W_{\text{temp}}^{-3}; \]

\[ Z(1,5) = W_{\text{temp}}^{-4}; \]

\[ W(1,1) = Z(1,1); \]

\[ W(1,2) = Z(1,2); \]

\[ W(1,3) = Z(1,3); \]

\[ W(1,4) = Z(1,4); \]

\[ W(1,5) = Z(1,5); \]

\[
\text{%%----------------------------------------------------------------------}}
\text{%% Calculation of}
\text{%% \[ E \cdot Z(I) \cdot W(J) + A1 \cdot A2 - A1 \cdot Z(I) - A2 \cdot W(J) \]}
\text{%% For all \((I,J) \leq (2,2)\)}
\text{%%----------------------------------------------------------------------}}
\]

\[ \text{for } i=1:5 \]
\[ \text{\quad for } j=1:5 \]
\[ K_{i,j} = E \cdot Z(i) \cdot Z(i) \cdot W(j) \cdot W(j) - A0 \cdot Z(i) \cdot W(j) - A1 \cdot Z(i) - A2 \cdot W(j); \]
\[ \text{\quad end} \]
\[ \text{end} \]

\[ \text{for } i=1:5 \]
\[ \text{\quad for } j=1:5 \]

62
str=strcat('K(',num2str(i-1),','_,num2str(j-1),')=');
disp(str)
disp(K{i,j}) %Display K
end

%%----------------------------------------------------------------------%%
%% Calculation of A(IJ) %%
%%----------------------------------------------------------------------%%
for i=1:5
  for j=1:5
    A(i,j)=det(K{i,j});
  end
end

%GA=num2cell(A)

%%----------------------------------------------------------------------%%
%% Calculation of F(IJ) %%
%%----------------------------------------------------------------------%%
for i=1:5
  for j=1:5
    F(i,j)=C*inv(K{i,j})*A(i,j)*(B1*Z(i)+B2*W(j));
  end
end
end

\%\%---------------------------------------------------------------------\%
\%\% Calculation of \%\%
\%\% P(IJ) \%\%
\%\%---------------------------------------------------------------------\%

P=zeros(5,5);
for lambda1=1:5
    for lambda2=1:5
        for i=1:5
            for j=1:5
                P(lambda1,lambda2)= P(lambda1,lambda2) ...
                  .+ F(i,j)*Wtemp^((i-1)*(lambda1-1)+(j-1)*(lambda2-1));
            end
        end
        P(lambda1,lambda2)=round(P(lambda1,lambda2))/25;
    end
end

P=lambda1=lambda2=round(P(lambda1,lambda2))/25;

end
end

\%\%---------------------------------------------------------------------\%
\%\% Calculation of \%\%
\%\% Q(IJ) \%\%
\%\%---------------------------------------------------------------------\%
Q=zeros(5,5);
for lambda1=1:5
    for lambda2=1:5
        for i=1:5
            for j=1:5
                Q(lambda1,lambda2)= Q(lambda1,lambda2)...
                .+ A(i,j)*Wtemp^((i-1)*(lambda1-1)+(j-1)*(lambda2-1));
            end
        end
        Q(lambda1,lambda2)=round(Q(lambda1,lambda2))/25;
    end
end
end
A %Display A
F %Display F
P %Display Numerator Polynomial
Q %Display Denominator Polynomial
Calculating the Transfer Function using formula:

\[ T(z_1, z_2) = C \cdot ((I \cdot (z_1^2 \cdot z_2^2) - A_0 \cdot z_1 \cdot z_2 - A_1 \cdot z_1 - A_2 \cdot z_2)^{-1}) \cdot (B_1 \cdot z_1 + B_2 \cdot z_2) \]

for \( z_1 = 1:5 \)

for \( z_2 = 1:5 \)

\[ \text{Transfer}(z_1, z_2) = C \cdot ((I \cdot (z_1^2 \cdot z_2^2) - A_0 \cdot z_1 \cdot z_2 - A_1 \cdot z_1 - A_2 \cdot z_2)^{-1}) \cdot (B_1 \cdot z_1 + B_2 \cdot z_2) \]

end

end

Transfer %Display Transfer function

Graphing the A matrix

\( w_1 = -6.8; \)
\( w_2 = 6.8; \)
\( \text{step} = 0.1; \)
\( [w_1, w_2] = \text{meshgrid}(w_1: \text{step}: w_2, w_1: \text{step}: w_2); \)
\( \text{plot} \cdot \text{real}(A), \text{imag}(A) \)
\( \text{subplot}(2, 2, 1) \)
\( \text{meshc}(A) \)
\( \text{axis square} \)
\( \text{shading interp} \)
title ('Fig X.XX transfer function mesh plot with contours')
subplot(2,2,2)
contour(A)
axis square
shading interp
title ('Fig X.XX transfer function contour plot')
subplot(2,2,3)
surf(A)
axis square
shading interp
title ('Fig X.XX transfer function surf plot')
subplot(2,2,4)
contourf(A,30)
axis square
title ('Fig X.XX transfer function contour plot of peaks')
pause(4)

%%-------------------------------------------------------------------%%
% Graphing the F matrice %
%%-------------------------------------------------------------------%%

ww1 = -6.8;
ww2 = 6.8;
step = 0.1;
[w1,w2]=meshgrid(ww1:step:ww2, ww1:step:ww2);
plot(real(F), imag(F))
subplot(2,2,1)
meshc(F)
axis square
shading interp
title ('Fig X.XX transfer function mesh plot with contours')
subplot(2,2,2)
contour(F)
axis square
shading interp
title ('Fig X.XX transfer function contour plot')
subplot(2,2,3)
surf(F)
axis square
shading interp
title ('Fig X.XX transfer function surf plot')
subplot(2,2,4)
contourf(F,30)
axis square
title ('Fig X.XX transfer function contour plot of peaks')

%%%---------------------------------------------------------------------%%%
MATLAB™ code for the figures in the thesis.

clear;
pack;
clg;

ww1 = -4.8;
ww2 = 4.8;
step = 0.1;
[w1,w2]=meshgrid(ww1:step:ww2, ww1:step:ww2);
j=sqrt(-1);
z1=exp(j*w1);
z2=exp(j*w2);

%%%% Example-1 data with 1 input and 1 output %

z_A= (z1.*z2+2*z2)...;
%./(z1.*z1.*z1.*z1.*z2.*z2.*z2.*z2 - z1.*z1.*z1.*z2.*z2.*z2 - z1.*z1.*z2.*z2.*z2 + z1.*z1.*z2 - z1.*z1 - 2*z1.*z2.*z2 - 2*z1.*z2 - z2.*z2);
% Example-2 data with 2 input and 1 output

\[ z_{B,\text{input1}} = \frac{(z1.*z1.*z1.*z1.*z2.*z2 + 3*z1.*z1.*z2.*z2.*z2 - 9*z1.*z1.*z1 - 4*z1.*z1 - 7*z1.*z2 - 3*z2.*z2)...}{(z1.*z1.*z1.*z1.*z2.*z2.*z2.*z2 - z1.*z1.*z1.*z2.*z2.*z2 - z1.*z1.*z2.*z2.*z2 + z1.*z1.*z2 - z1.*z1 - 2*z1.*z2.*z2 - 2*z1.*z2 - z2.*z2)}; \]

\[ z_{B,\text{input2}} = \frac{(2*z1.*z1.*z1.*z2.*z2 + z1.*z1.*z2.*z2.*z2 - 2*z1.*z1 + 3*z1.*z2.*z2 - 2*z1.*z2)...}{(z1.*z1.*z1.*z1.*z2.*z2.*z2.*z2 - z1.*z1.*z1.*z2.*z2.*z2 - z1.*z1.*z2.*z2.*z2 + z1.*z1.*z2 - z1.*z1 - 2*z1.*z2.*z2 - 2*z1.*z2 - z2.*z2)}; \]

% Example-3 data with 2 input and 2 output

\[ z_{C,\text{input1_output1}} = \frac{(3*z2.*z2 + z1.*z2 - 3*z1.*z1 + 3*z1.*z1.*z2 - 3*z1.*z1.*z1.*z2.*z2)...}{(z1.*z1.*z1.*z1.*z2.*z2.*z2.*z2 - z1.*z1.*z1.*z2.*z2.*z2 - z1.*z1.*z2.*z2.*z2 + z1.*z1.*z2 - z1.*z1 - 2*z1.*z2.*z2 - 2*z1.*z2 - z2.*z2)}; \]

\[ z_{C,\text{input1_output2}} = \frac{(z2.*z2 + 3*z1.*z2 - z1.*z2.*z2 + z1.*z1.*z2.*z2.*z2)...}{(z1.*z1.*z1.*z1.*z2.*z2.*z2.*z2 - z1.*z1.*z1.*z2.*z2.*z2 - z1.*z1.*z2.*z2.*z2 + z1.*z1.*z2 - z1.*z1 - 2*z1.*z2.*z2 - 2*z1.*z2 - z2.*z2)}; \]

\[ z_{C,\text{input2_output1}} = \frac{(-3*z2.*z2 - 13*z1.*z2 - 11*z1.*z1 - 15.*z1.*z1.*z2 + 6*z1.*z1.*z2.*z2.*z2 - z1.*z1.*z1.*z2.*z2.*z2)...}{(z1.*z1.*z1.*z1.*z2.*z2.*z2.*z2 - z1.*z1.*z1.*z2.*z2.*z2 - z1.*z1.*z2.*z2.*z2 + z1.*z1.*z2 - z1.*z1 - 2*z1.*z2.*z2 - 2*z1.*z2 - z2.*z2)}; \]

\[ z_{C,\text{input2_output2}} = \frac{(z2.*z2 + 3*z1.*z2 - z1.*z2.*z2 + z1.*z1.*z2.*z2.*z2)...}{(z1.*z1.*z1.*z1.*z2.*z2.*z2.*z2 - z1.*z1.*z1.*z2.*z2.*z2 - z1.*z1.*z2.*z2.*z2 + z1.*z1.*z2 - z1.*z1 - 2*z1.*z2.*z2 - 2*z1.*z2 - z2.*z2)\text{...;}} \]
\[-z_1.*z_2.*z_2.*z_2 + z_1.*z_1.*z_2 - z_1.*z_1 - 2*z_1.*z_2.*z_2 - 2*z_1.*z_2 - z_2.*z_2);\]

\[z_{C\_input2\_output2}= (z_2.*z_2 - z_1.*z_2 + 5*z_1.*z_2.*z_2 - 4*z_1.*z_1 + 3*z_1.*z_2.*z_2.*z_2 + 4*z_1.*z_1.*z_2.*z_2)...;\]

\.[/(z_1.*z_1.*z_1.*z_1.*z_2.*z_2.*z_2.*z_2 - z_1.*z_1.*z_1.*z_2.*z_2 - z_1.*z_1.*z_2.*z_2 + z_1.*z_1.*z_2 - z_1.*z_1 - 2*z_1.*z_2.*z_2 - 2*z_1.*z_2 - z_2.*z_2);\]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Plotting the transfer function of the Single-Input Single-Output example%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

subplot(2,2,1), meshc(abs(z_A));
grid on;
title('ex. A Original magnitude ');

subplot(2,2,2),contour(abs(z_A));
grid on;
title('ex. A Contour of original mag. '); 

subplot(2,2,3),surf(abs(z_A),'facecolor','black','edgecolor','yellow');
grid on;
colormap hsv
shading interp 
title('ex. A surface of original mag. '');

subplot(2,2,4), plot(real(z_A), imag(z_A), 'o');
grid on;
title('ex. A z plot of original mag. ');
```matlab
% Plotting the transfer function of the two-Input Single-Output example

% Plotting: Input1

figure

subplot(2,2,1), meshc(abs(z_B_input1));
grid on;
title('ex. B Input 1 - Original magnitude ');

subplot(2,2,2), contour(abs(z_B_input1));
grid on;
title('ex. B Input 1 - Contour of original mag.');

subplot(2,2,3), surf(abs(z_B_input1), 'facecolor', 'black', 'edgecolor', ...
                    'yellow');
grid on;
colormap hsv
shading interp
title('ex. B Input 1 - surface of original mag.');

subplot(2,2,4), plot(real(z_B_input1), imag(z_B_input1), 'o');
grid on;
title('ex. B Input 1 - z plot of original mag.');
pause(4)
```
% Plotting the transfer function of the two-Input Single-Output example  
% Plotting: Input2

subplot(2,2,1), meshc(abs(z_B_input2));
grid on;
title('ex. B Input 2 - Original magnitude ');

subplot(2,2,2), contour(abs(z_B_input2));
grid on;
title('ex. B Input 2 - Contour of original mag.');

subplot(2,2,3), surf(abs(z_B_input2),'facecolor','black','edgecolor',
                   '.yellow');
grid on;
colormap hsv
shading interp

title('ex. B Input 2 - surface of original mag. ');

subplot(2,2,4), plot(real(z_B_input2), imag(z_B_input2), 'o');
grid on;
title('ex. B Input 2 - z plot of original mag.');
pause(4)

% Plotting the transfer function of the Two-Input Two-Output example  
% Plotting: Input1 - Output1

subplot(2,2,1), meshc(abs(z_C_input1_output1));
grid on;
title('ex. C Input 1 - Output 1 - Original magnitude ');

subplot(2,2,2),contour(abs(z_C_input1_output1));
grid on;
title('ex. C Input 1 - Output 1 - Contour of original mag.');

subplot(2,2,3),surf(abs(z_C_input1_output1),'facecolor','black',...
        .'edgecolor','yellow');
grid on;
colormap hsv
shading interp
title('ex. C Input 1 - Output 1 - surface of original mag.');

subplot(2,2,4),plot(real(z_C_input1_output1),imag(z_C_input1_output1),'o');
grid on;
title('ex. C Input 1 - Output 1 - z plot of original mag.');
pause(4)

% Plotting the transfer function of the Two-Input Two-Output example  
% Plotting: Input1 - Output2

subplot(2,2,1), meshc(abs(z_C_input1_output2));
grid on;
title('ex. C Input 1 - Output 2 - Original magnitude ');

subplot(2,2,2),contour(abs(z_C_input1_output2));
grid on;
title('ex. C Input 1 - Output 2 - Contour of original mag.');
```matlab
subplot(2,2,3), surf(abs(z_C_input1_output2), 'facecolor', 'black',...
    , 'edgecolor', 'yellow');
grid on;
colormap hsv
shading interp
title('ex. C Input 1 - Output 2 - surface of original mag.');

subplot(2,2,4), plot(real(z_C_input1_output2),imag(z_C_input1_output2),'o');
grid on;
title('ex. C Input 1 - Output 2 - z plot of original mag.');
pause(4)

% Plotting the transfer function of the Two-Input Two-Output example
% Plotting: Input2 - Output1

subplot(2,2,1), meshc(abs(z_C_input2_output1));
grid on;
title('ex. C Input 2 - Output 1 - Original magnitude ');

subplot(2,2,2), contour(abs(z_C_input2_output1));
grid on;
title('ex. C Input 2 - Output 1 - Contour of original mag.');

subplot(2,2,3), surf(abs(z_C_input2_output1), 'facecolor', 'black',...
    , 'edgecolor', 'yellow');
grid on;
colormap hsv
shading interp
title('ex. C Input 2 - Output 1 - surface of original mag.');
```
subplot(2,2,4), plot(real(z_C_input2_output1), imag(z_C_input2_output1), 'o');
grid on;
title('ex. C Input 2 - Output 1 - z plot of original mag.');
pause(4)

% Plotting the transfer function of the Two-Input Two-Output example
% Plotting: Input2 - Output2

subplot(2,2,1), meshc(abs(z_C_input2_output2));
grid on;
title('ex. C Input 2 - Output 2 - Original magnitude');

subplot(2,2,2), contour(abs(z_C_input2_output2));
grid on;
title('ex. B Input 2 - Output 2 - Contour of original mag.');

subplot(2,2,3), surf(abs(z_C_input2_output2), 'facecolor', 'black', 'edgecolor', 'yellow');
grid on;
colormap hsv
shading interp

title('ex. B Input 2 - Output 2 - surface of original mag.');

subplot(2,2,4), plot(real(z_C_input2_output2), imag(z_C_input2_output2), 'o');
grid on;
title('ex. B Input 2 - Output 2 - z plot of original mag.');
REFERENCES


Marinos T. Michael was born in Limassol, Cyprus. He was awarded his degree of Bachelor of Science in Computer Science (Magna Cum Laude) in 2004 and the degree of Master of Science in Computer Science in 2006 from Montclair State University, Montclair, New Jersey, USA. He is a student member of the IEEE.