Instituto Tecnológico de Costa Rica

Escuela de Matemáticas

Proyecto de Investigación

5402-1440-4301

Transformadas Discretas de Fourier: Un marco matemático-computacional para el desarrollo de algoritmos en paralelo

---

Informe Final

Autores:

M.Sc. Geisel Alpízar Brenes

M.Sc. Cindy Calderón Arce

M.Sc. Juan Pablo Soto Quirós

2015

Índice

1. Datos Generales

1.1. Nombre del proyecto

- Transformadas Discretas de Fourier: Un marco matemático-computacional para el desarrollo de algoritmos en paralelo.

1.2. Código

- # 5402-1440-4301

1.3. Autores y direcciones

- M.Sc. Geisel Alpízar Brenes: Escuela de Matemáticas (galpizar@tec.ac.cr)
- M.Sc. Cindy Calderón Arce: Escuela de Matemáticas (ccalderon@tec.ac.cr)
- M.Sc. Juan Pablo Soto Quirós: Escuela de Matemáticas (jusoto@tec.ac.cr)

1.4. Resumen

Este proyecto presenta un marco matemático-computacional para el desarrollo de un conjunto de definiciones derivadas de la transformada discreta de Fourier (TDF), que son la función discreta de ambigüedad, la transformada discreta de Zak, la transformada discreta de Fourier en tiempo corto, la transformada discreta chirp-Fourier, la transformada discreta de Fourier de quaterniones, la transformada discreta de Cohen, la transformada hipercompleja discreta de Fourier y la transformada discreta de Fourier de valores vectoriales.

Se entenderá como marco computacional matemático al conjunto formado por una formulación matemática de un algoritmo y su implementación en algún lenguaje de programación. Este marco matemático-computacional se desarrolla a través de un álgebra matricial de señales, el cual consiste en un ambiente matemático compuesto de un conjunto de espacios de señales, operadores lineales y un conjunto de matrices especiales, donde los métodos algebraicos se utilizan para generar señales que se transforman como estimadores computacionales. Además, el álgebra matricial de señales contribuye al análisis, diseño e implementación de algoritmos en paralelo; por lo tanto, cada una de las formulaciones matemáticas de las definiciones de la TDF presentarán una representación que permitirá su cómputo en paralelo.

El lenguaje de programación a utilizar para implementar cada uno de los algoritmos de las definiciones derivadas de la TDF es MATLAB utilizando el Parallel Computing Toolbox. La implementación de estos algoritmos en MATLAB permite aprovechar el uso de procesadores multinúcleo, al asignar el cómputo de una instancia independiente en cada procesador y mejorar el cómputo de estas transformadas.

1.5. Palabras Clave

Transformada discreta de Fourier, transformada discreta de Zak, transformada discreta de Fourier en tiempo corto, transformada discreta chirp-Fourier, transformada fraccionaria discreta de Fourier, transformada discreta Clifford-Fourier, transformada hipercompleja discreta de Fourier, transformada discreta de Fourier de valores vectoriales, álgebra matricial de señales, cómputo en paralelo.

2. Introducción

La transformada discreta de Fourier (TDF) es considerada una de las principales herramientas que permite la conversión entre el tiempo y la representación de dominio de frecuencia de señales discretas finitas [15, 36]. La TDF es la principal transformada que se utiliza en el análisis discreto de Fourier (ADF) [41]. La TDF tiene diversas aplicaciones en el procesamiento de señales: tratamiento de señales de audio e imágenes [23, 36], reducción de ruido en señales [43], análisis en frecuencia de señales discretas [30], el análisis de materiales [15], entre otros. También, se utiliza en diversas áreas de la matemática: análisis armónico numérico [8], teoría de grupos [11], álgebra lineal y teoría de matrices [36], entre otros. Además, la implementación de la TDF se puede hacer en una forma eficiente y realizando cómputo en paralelo, utilizando una representación matricial, a través de la transformada rápida de Fourier [17, 36].

A partir de la definición original de la TDF se han derivado múltiples definiciones alternas de la TDF, las cuales presentan alguna modificación de la definición original y la cual tiene una aplicación específica en alguna área de investigación en la matemática y en la ingeniería. Las que se consideran en estre proyecto de investigación son las siguientes:

- la transformada discreta discreta de Fourier de quaterniones (TDFQ) [14],
- la transformada discreta de Fourier de valores vectoriales (TDFVV) [5, 29],
- la función discreta de ambigüedad (FDA) [4],
- la transformada discreta de Zak (TDZ) [7],
- la transformada discreta de Fourier en tiempo corto (TDFTC) [24],
- la transformada discreta chirp-Fourier (TDChF) [42] y
- la transformada discreta de Cohen (TDC) [5].

A este conjunto de transformadas se les conocerá como transformadas en el ADF. Las aplicaciones de estas transformadas son diveras. Algunas de estas aplicaciones se encuentran en la teoría de las expansiones de Gabor y marcos de Weyl-Heisenberg (FDA y TDZ, ver, por ejemplo, [3, 16]), en la representación en tiempo y frecuencia de señales para eliminación de ruido (FDA, TDFTC y TDC, ver, por ejemplo, [19]), en canales de múltiple entrada y múltiple salida en los sistemas radar de apertura sintética (TDChF y TDFVV, ver por ejemplo, [14]), procesamiento de imágenes (TDFQ, ver, por ejemplo, [14]), entre otros.

Este proyecto presenta el desarrollo de un marco matemático-computacional para cada una de las definiciones derivadas de la TDF. Se entenderá como marco computacional matemático al conjunto formado por una formulación matemática de un algoritmo y su implementación en

algún lenguaje de programación. Este marco matemático-computacional se desarrolla a través de un álgebra matricial de señales, el cual consiste en un ambiente matemático compuesto de un conjunto de espacios de señales, operadores lineales y un conjunto de matrices especiales, donde los métodos algebraicos se utilizan para generar señales que se transforman como estimadores computacionales. Además, el álgebra matricial de señales contribuye al análisis, diseño e implementación de algoritmos en paralelo; por lo tanto, cada una de las formulaciones matemáticas de las definiciones de la TDF presentarán una representación que permitirá su cómputo en paralelo.

Las transformadas mencionadas anteriormente no son las únicas transformadas que se utilizan en el ADF. Por ejemplo, también se encuentran la transformada discreta *wavelet* [40] y la transformada discreta fraccionaria de Fourier [10]. En este proyecto no se consideran dichas transformadas porque su computo en paralelo usando representación matricial es diferente al enfoque presentado en este informe.

El lenguaje de programación a utilizar para implementar cada uno de los algoritmos de las definiciones derivadas de la TDF es MATLAB. Para ello, se utilizan una herramienta computacional de MATLAB: *Parallel Computing Toolbox* [22], el cual permite resolver problemas computacionales y de datos intensivos que utilizan procesadores multinúcleo. El objetivo principal en la parte de programación es elaborar un conjunto de funciones en MATLAB de las definiciones derivadas de la TDF y agruparlas mediante un nuevo toolbox. Dichas funciones permitirán graficar cada una de las transformadas teniendo como parámetro una señal dada. Además, se desarrolla una interfaz gráfica para el usuario, a través del entorno de desarrollo que viene por defecto en MATLAB [21]. La importancia de desarrollar algoritmos que permiten su cómputo en paralelo recae en el hecho de que la mayoría de las computadoras, en la actualidad, poseen procesadores de múltiples núcleos. La implementación de estos algoritmos en MATLAB utilizando el *Parallel Computing Toolbox* permite aprovechar dichos procesadores, al asignar el cómputo de una instancia independiente en cada procesador y acelear el cómputo de estas transformadas.

3. Marco Teórico

3.1. Notación

En este documento se utilizará la siguiente notación: Sea $\mathbb{S}^{m\times n}$ el espacio de matrices de m filas y n columnas con entradas en un conjunto no vacío $\mathbb{S}$, donde $\mathbb{S}^{n\times 1} = \mathbb{S}^n$ es el espacio de vectores de dimensión n con entradas en $\mathbb{S}$. Las letras mayúsculas en negritas, por ejemplo $\mathbf{A}$, representará a las matrices, y las letras minúsculas en negritas, por ejemplo $\mathbf{x}$, representa a los vectores. Las filas y columnas de $\mathbf{A} \in \mathbb{S}^{m\times n}$ son indexadas con los espacios $\mathbb{Z}_m$ y $\mathbb{Z}_n$, respectivamente. $\mathbf{A}^T$ y $\overline{\mathbf{A}}$ representa a la transpuesta y conjugada de la matriz $\mathbf{A}$. $\mathbf{A}(r, s)$, $\mathbf{A}(r, :)$ y $\mathbf{A}(:, s)$ representa la entrada (r, s), fila r y columna s de la matriz $\mathbf{A}$, respectivamente. Sea $\mathbf{I}_n$ y $\mathbf{1}_n$ la matriz identidad de dimensión $n \times n$ y el vector de entradas 1 en $\mathbb{C}^n$, es decir, $\mathbf{1}_n(k) = 1$, para todo $k \in \mathbb{Z}_n$, respectivamente.

3.2. Análisis discreto de Fourier

El análisis de Fourier es el área de la matemática en la cual se estudia cómo ciertas funciones pueden ser descompuestas en funciones trigonométricas o exponenciales con frecuencias definidas [41]. En procesamiento de señales, este tipo de funciones se llaman señales. Las señales se dividen en dos categorías: *las señales de tiempo continuo* y *señales de tiempo discreto* [15]. En el análisis de Fourier, las transformaciones más populares para señales continuas son las transformadas de Fourier. Una implementación computacional de transformadas de Fourier se realiza usando las señales de tiempo discreto y su transformada discreta, conocida como transformada discreta de Fourier (TDF). El campo de investigación que abarca el estudio de las señales de tiempo discreto y las transformadas discretas de Fourier se conoce como *análisis discreto de Fourier* [41].

El análisis discreto de Fourier tiene varias aplicaciones en ingeniería, especialmente en el procesamiento de imágenes y señales. Por ejemplo, en la compresión de la señal y la eliminación de ruido, el análisis espectral, bancos de filtros, tomografía, los sistemas de micro-electromecánicos y biomecánica computacionales [9, 41]. En todas estas aplicaciones de ingeniería, en las señales de tiempo continuo se obtienen muestras a intervalos de tiempo discretos antes de ser procesada. Dichas muestras dan forma a las señales de tiempo discreto.

3.3. Transformadas Discretas de Fourier

Sea $\mathbb{Z}_n = \{0, 1, 2, ..., n-1\}$ y $\mathbb{S}$ un espacio no vacío. Una señal de tiempo discreto en una dimensión es una función $\mathbf{x}$ con dominio $\mathbb{Z}_n$ y codominio $\mathbb{S}$, tal que

$$\begin{aligned} \mathbf{x}: \ \mathbb{Z}_n &\to \mathbb{S} \\ j &\to \mathbf{x}(j) \end{aligned}$$

para todo $j \in \mathbb{Z}_n$ [41]. El espacio de estas señales se denota con el símbolo $l^2(\mathbb{Z}_n, \mathbb{S})$. De forma similar, una señal de tiempo discreto en dos dimensiones es una función $\mathbf{X}$ con dominio $\mathbb{Z}_m \times \mathbb{Z}_n$ y codominio $\mathbb{S}$, tal que

$$\begin{aligned} \mathbf{X}: \ \mathbb{Z}_m \times \mathbb{Z}_n &\to \mathbb{S} \\ (j,k) &\to \mathbf{X}(j,k) \end{aligned}$$

para todo $j \in \mathbb{Z}_m$ y $k \in \mathbb{Z}_n$ [41]. El espacio de estas señales se denota con el símbolo $l^2(\mathbb{Z}_m \times \mathbb{Z}_n, \mathbb{S})$. De lo anterior, se deduce que $l^2(\mathbb{Z}_n, \mathbb{S}) = \mathbb{S}^n$ y $l^2(\mathbb{Z}_m \times \mathbb{Z}_n, \mathbb{S}) = \mathbb{S}^{m \times n}$ [4]. Las dos transformadas discretas de Fourier más populares son *la transformada discreta de Fourier de una dimensión* (TDF-1D) y *la transformada discreta de Fourier de dos dimensiones* (TDF-2D).

- Sea $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{S})$. La TDF-1D se define como la aplicación $\mathcal{F}: \mathbb{Z}_n \to \mathbb{S}$, tal que

  $$\mathcal{F}(k) = \sum_{j \in \mathbb{Z}_n} \omega_n^{-jk} \mathbf{x}(k), \quad \forall k \in \mathbb{Z}_n,$$

  donde ω_n se conoce como núcleo. El valor de ω_n depende del conjunto $\mathbb{S}$. En este proyecto se consideran tres casos para el conjunto $\mathbb{S}$:

  - Si $\mathbb{S} = \mathbb{C}$, entonces $\omega_n = e^{i\frac{2\pi}{n}}$, donde $i^2 = -1$. Esta transformada se conoce como TDF de valores complejos, o simplemente TDF [36].
  - Si $\mathbb{S} = \mathbb{H}$, donde $\mathbb{H}$ es el espacio de cuaterniones, entonces $\omega_n = e^{\mu\frac{2\pi}{n}}$, donde μ es un cuaternion puro. Esta transformada se conoce como TDF de cuaterniones [14].
  - Si $\mathbb{S} = \mathbb{C}^d$, entonces $\omega_n \in \mathbb{C}^{d \times d}$. Esta transformada se conoce como la TDF de valores vectoriales [5, 29].

- Sea $\mathbf{X} \in l^2(\mathbb{Z}_m \times \mathbb{Z}_n, \mathbb{S})$. La TDF-2D se define como la aplicación $\mathfrak{F}: \mathbb{Z}_m \times \mathbb{Z}_n \to \mathbb{S}$, tal que

  $$\mathfrak{F}(j,k) = \sum_{r \in \mathbb{Z}_m} \sum_{s \in \mathbb{Z}_n} \omega_m^{-jr} \omega_n^{-sk} \mathbf{X}(r,s),$$

  para todo $j \in \mathbb{Z}_m$ y $k \in \mathbb{Z}_n$. El valor de ω_n se define de forma similar a la TDF-1D, y cuando $\mathbb{S} = \mathbb{C}^d$, entonces se define si y solo si $m = n$.

3.4. Transformadas en tiempo-discreto y frecuencia-discreta

La TDF-1D y TDF-2D se utilizan para el análisis de frecuencia de las señales de tiempo discreto, y por lo tanto, son conocidas como transformadas discretas de frecuencias. Desde el punto de vista del análisis de tiempo-discreto y discreta-frecuencia, una limitación de las TDF es que no incluyen información de la ubicación en el tiempo, sólo información de la frecuencia, y por lo tanto tienen dificultad en la representación de los transitorios. Una alternativa para este problema son las transformadas de tiempo-discreto y discreta-frecuencia (TD-FD) [12, 20, 26]. Para $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{S})$, una transformada en TD-FD se define como

$$\begin{array}{rccc} \mathcal{T}_{\mathbf{x}}: & \mathbb{Z}_m \times \mathbb{Z}_n & \to & \mathbb{S} \\ & (j,k) & \to & \mathcal{T}_{\mathbf{x}}(j,k). \end{array}$$

Las transformadas en TD-FD pueden ser calculadas usando la TDF-1D y la TDF-2D. Las transformaciones de tiempo-discreto y análisis de dominio discreto-frecuencia se combinan para producir una imagen de la localización temporal de los componentes de señales espectrales [20]. Según la literatura consultada [4, 5, 6, 7, 24, 42], las transformadas en TD-FD se definen solo para el caso $\mathbb{S} = \mathbb{C}$ y se pueden dividir en dos categorías:

Categoría 1: Sea $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{C})$, la transformada en TD-FD de categoría 1 $\mathcal{T}_{\mathbf{x}} : \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{C}$ se define como

$$\mathcal{T}_{\mathbf{x}}(m,k) = \mathbf{A}(m,k) \sum_{r \in \mathbb{Z}_n} \mathbf{x}(r)\mathbf{H}(m,r)\omega_n^{-rk},$$

donde $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{C})$ y $\mathbf{A}, \mathbf{H} \in \mathbb{C}^{n \times n}$ son dados en la Tabla 1. Este tipo de transformadas definen a la función discreta de ambigüedad (FDA), la transformada discreta de Zak (TDZ), la transformada discreta de Fourier en tiempo corto (TDFTC), la transformada discreta chirp-Fourier (TDCF) y sus derivadas: la transformada discreta chirp-Fourier modificada (TDCFM) y nueva la transformada discreta chirp-Fourier (NTDCF). Para el cómputo de la FDA, se utiliza una señal adicional $\mathbf{y}$ conocida como *señal eco* [2] y para el cómputo de la transformada discreta de Fourier en tiempo corto se utiliza una señal adicional $\mathbf{w}$ conocida como *señal ventana* [6].

Transformadas en TD-FD	$\mathbf{A}(m,k)$	$\mathbf{H}(m,r)$
FDA	1	$\overline{\mathbf{y}}(r+m)$
TDFTC	1	$\mathbf{w}(r-m)$
TDZ	ω_r^{mk}	1
TDCF	1	$\omega_n^{-mr^2}$
TDCFM	1	$(1+(-1)^r)\,\omega_n^{-mr^2/2}$
NTDCF	$(-1)^k$	$\omega_n^{-m(r-n/2)^2/2}$

Cuadro 1: Valores de $\mathbf{A}$ y $\mathbf{H}$ de la transformada en TD-FD de categoría 1

Categoría 2: Sean $\mathbf{x}, \mathbf{y} \in l^2(\mathbb{Z}_n, \mathbb{C})$, la transformada en TD-FD de categoría 2, también conocida como transformada discreta de Cohen (TDC) $\mathcal{C}_{\mathbf{x},\mathbf{y}} : \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{C}$ es definida como

$$\mathcal{C}_{\mathbf{x},\mathbf{y}}(j,k) = \sum_{r\in\mathbb{Z}_n} \sum_{s\in\mathbb{Z}_n} \mathbf{A}_{\mathbf{x},\mathbf{y}}(r,s)\Phi(r,s)\omega_n^{-(ms+kr)},$$

donde $\mathbf{A}_{x,y}$ representa a la función discreta de ambigüedad de $\mathbf{x}$, con función eco $\mathbf{y}$, y $\Phi \in \mathbb{C}^{n\times n}$ es el núcleo de TDC, el cual es dado en la Tabla 2.

Núcleo	$\Phi(r,s)$
Wigner	1
Margenau - Hill	$\cos(\pi rs)$
Kirwood-Rihanzek	$e^{-\pi rs}$
Born - Jordan	$\frac{\text{sen}(\pi rs)}{\pi rs}$
Choi - Williams	$e^{-\alpha(\pi r\theta/2)^2}$
Zhao-Atlas-Marks	$e^{-\alpha r^2}\,\lvert r\rvert\,\frac{\text{sen}(\alpha\pi\theta r/2)}{\alpha\pi\theta r/2}$

Cuadro 2: Valores de Φ de la transformada en TD-FD de categoría 2

4. Problema y Objetivos

4.1. Problema

La investigación plantea como problema el siguiente: *Generar un marco computacional-matemático para un conjunto de definiciones derivadas de la transformada discreta de Fourier utilizando un álgebra matricial de señales.*

4.2. Objetivos específicos

1. Desarrollar formulaciones matemáticas utilizando el álgebra matricial de señales para cada una de las transformadas discretas en el análisis discreto de Fourier.

2. Implementar cada una de las transformadas discretas en MATLAB utilizando cómputo en paralelo.

3. Probar cada una de las transformadas implementadas en MATLAB.

4. Elaborar una interface gráfica para el manejo de las funciones implementadas en MATLAB.

5. Elaborar una guía de usuario de la aplicación desarrollada en MATLAB.

5. Metodología

Para lograr los objetivos del proyecto, éste fue desarrollado por etapas.

5.1. Álgebra matricial de señales y las TDF

5.1.1. Álgebra matricial de señales

En esta primera etapa se define el concepto de un álgebra matricial de señales, la cual será la base fundamental para el desarrollo del marco computacional, el cual se implementa en paralelo.

El álgebra matricial de señales es un ambiente matemático compuesto de un espacio de señales, operadores lineales finitos y con conjunto especial de matrices, donde métodos algebraicos son usados para generar algoritmos en el área de procesamiento de señales. En este proyecto, los espacios de señales a considerar son $l^2(\mathbb{Z}_n, \mathbb{S})$ y $l^2(\mathbb{Z}_m \times \mathbb{Z}_n, \mathbb{S})$. Este tipo de álgebra contribuye al análisis, diseño e implementación de algoritmos en paralelo. A continuación, se presenta el conjunto de operadores y matrices que se utilizan del álgebra matricial de señales[1]. Sean $\mathbf{A}, \mathbf{B} \in \mathbb{S}^{m\times n}$, $\mathbf{C} \in \mathbb{S}^{p\times q}$ y el conjunto de matrices $\{\mathbf{A}_r\}_{r\in\mathbb{Z}_k}$ tales que $\mathbf{A}_r \in \mathbb{S}^{m_r\times n_r}$:

- *El producto Hadamard*, también conocido como multiplicación entrada por entrada, de $\mathbf{A}$ y $\mathbf{B}$ se define como $\mathbf{A}\odot\mathbf{B} \in \mathbb{S}^{m\times n}$ tal que

$$(\mathbf{A}\odot\mathbf{B})(r,s) = \mathbf{A}(r,s)\mathbf{B}(r,s),$$

  donde $\mathbf{A}(r,s)\mathbf{B}(r,s)$ es la multiplicación definida en $\mathbb{S}$.

- *El producto Kronecker* entre $\mathbf{A}$ y $\mathbf{C}$ es una multiplicación por bloques definida como $\mathbf{A}\otimes\mathbf{C} \in \mathbb{S}^{mp\times nq}$ tal que

$$\mathbf{A}\otimes\mathbf{C} = \begin{pmatrix} \mathbf{A}(0,0)\mathbf{C} & \cdots & \mathbf{A}(0,n-1)\mathbf{C} \\ \vdots & \ddots & \vdots \\ \mathbf{A}(m-1,0)\mathbf{C} & \cdots & \mathbf{A}(m-1,n-1)\mathbf{C} \end{pmatrix},$$

  donde $\mathbf{A}(r,s)\mathbf{C}$ es la multiplicación definida en $\mathbb{S}$. Un caso a considerar es cuando $\mathbf{A} = \mathbf{I}_n$, el cual se le conoce como operación en paralelo [36].

[1] El concepto de algebra de señales es usado en los siguientes arículos: [27, 28, 32]. Estos operadores son comunes en teoría de matrices aplicado a procesamiento de señales. Ver más detalles de los operadores en [13, 17, 18, 25, 31, 36, 39]

- *La suma directa* de matrices $\{\mathbf{A}_r\}_{r\in\mathbb{Z}_k}$ construye una matriz diagonal por bloques de un conjunto de matrices, tal que

$$\mathbf{A} = \bigoplus_{r\in\mathbb{Z}_k} \mathbf{A}_r = \begin{pmatrix} \mathbf{A}_0 & & \\ & \ddots & \\ & & \mathbf{A}_{k-1} \end{pmatrix},$$

  donde $\mathbf{A} \in \mathbb{C}^{p\times q}$, $p = \sum_{r\in\mathbb{Z}_k} m_r$ y $q = \sum_{r\in\mathbb{Z}_k} n_r$.

Sea $n = rs$. *La matriz de permutación de paso s* se define como $\mathbf{L}^n_s \in \mathbb{C}^{n\times n}$ tal que permuta los elementos de la señal $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{S})$ de la forma $jr + k \to ks + j$, para $j \in \mathbb{Z}_s$ y $k \in \mathbb{Z}_r$. Esta matriz es una matriz binaria, es decir, las entradas están compuestas por los valores 1 y 0. Esta matriz de permutación rige el flujo de información requerido al utilizar el producto Kronecker [36].

El operador vec $\mathcal{V} : \mathbb{S}^{m\times n} \to \mathbb{S}^{mn}$ transforma una matriz en un vector, ordenando cada columna encima de la otra. *El operador inverso vec* $\mathcal{R}_{m,n} : \mathbb{S}^{mn} \to \mathbb{S}^{m\times n}$ transforma un vector de dimensión mn en una matriz de dimensión $m \times n$.

El siguiente ejemplo ilustra cómo el álgebra matricial de señales contribuye a la implementación en paralelo de algunos algoritmos.

Ejemplo 1 Sea $\mathbf{A} \in \mathbb{C}^{r\times m}$, $\mathbf{x} \in l^2(\mathbb{Z}_{nr}, \mathbb{S})$ y $\mathbf{y} \in l^2(\mathbb{Z}_{mn}, \mathbb{S})$. Consideremos la operación $\mathbf{x} \odot (\mathbf{I}_n \otimes \mathbf{A})\mathbf{y}$. Esta operación se puede descomponer de la siguiente forma

$$\begin{pmatrix} \mathbf{x}_0 \\ \vdots \\ \mathbf{x}_{n-1} \end{pmatrix} \odot \begin{pmatrix} \mathbf{A} & & \\ & \ddots & \\ & & \mathbf{A} \end{pmatrix} \begin{pmatrix} \mathbf{y}_0 \\ \vdots \\ \mathbf{y}_{n-1} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_0 \odot \mathbf{A}\mathbf{y}_0 \\ \vdots \\ \mathbf{x}_{n-1} \odot \mathbf{A}\mathbf{y}_{n-1} \end{pmatrix},$$

donde $\mathbf{x}_j \in l^2(\mathbb{Z}_r, \mathbb{S})$ y $\mathbf{y}_j \in l^2(\mathbb{Z}_m, \mathbb{S})$. La operación matricial $\mathbf{x}\odot(\mathbf{I}_n\otimes\mathbf{A})\mathbf{y}$ se puede dividir en n operaciones $\mathbf{x}_j \odot \mathbf{A}\mathbf{y}_j$, para $j \in \mathbb{Z}_n$. La estructura de $\mathbf{x}\odot(\mathbf{I}_n\otimes\mathbf{A})\mathbf{y}$ permite una implementación en paralelo, porque cada operación $\mathbf{x}_j \odot \mathbf{A}\mathbf{y}_j$ es calculada independientemente.

5.1.2. Implementación en paralelo de las TDF para señales complejas

Para el caso $\mathbb{S} = \mathbb{C}$, existe implementación en paralelo de la TDF-1D y TDF-2D, utilizando el álgebra matricial de señales.

- Sea $\mathcal{F} = \mathbf{F}_n\mathbf{x}$ la representación matricial de la TDF-1D de $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{S})$, donde $\mathbf{F}_n \in \mathbb{S}^{n\times n}$ tal que $\mathbf{F}_n(j,k) = \frac{1}{\sqrt{n}}\omega_n^{-jk}$. Si $n = rs$, entonces $\mathbf{F}_n$ se puede factorizar como:

$$\mathbf{F}_n\mathbf{x} = \mathbf{L}^n_s(\mathbf{I}_r \otimes \mathbf{F}_s)\mathbf{L}^n_r\mathbf{T}^n_r(\mathbf{I}_s \otimes \mathbf{F}_r)\mathbf{L}^n_s\mathbf{x}. \tag{1}$$

- Sea $\mathbf{F_X} \in \mathbb{S}^{m\times n}$, tal que $\mathbf{F_X}(m,k) = \mathfrak{F}_{\mathbf{X}}(m,k)$, entonces $\mathbf{F_X}$ se puede factorizar como:

$$\mathbf{F_X} = \mathcal{R}_{m,n}\left\{\mathbf{L}_n^{mn}(\mathbf{I}_m \otimes \mathbf{F}_n)\mathbf{L}_m^{mn}(\mathbf{I}_n \otimes \mathbf{F}_m)\mathcal{V}\{\mathbf{X}\}\right\}. \tag{2}$$

Utilizando lo anterior, se desarrollará la formulación matemática haciendo uso del álgebra matricial de señales para cada una de las transormadas en el ADF mencionadas en este proyecto.

5.2. Implementación en MATLAB utilizando cómputo en paralelo

La implementación computacional de cada una de las transformadas se desarrolla utilizando el software MATLAB, por medio de *Parallel Computing Toolbox* y *workers*. Nosotros utilizamos múltiples *workers* en MATLAB en una computadora multi-núcleo para ejecutar aplicaciones en paralelo. Este enfoque permite un mayor control sobre el paralelismo, utilizando bucles en paralelo (`parfor`).

MATLAB incorpora librerías para facilitar al usuario la programación de algoritmos que sean capaces de correr en paralelo utilizando procesadores multinúcleo, GPUs o clústers de ordenadores. Muchas de las funciones de optimización incorporadas pueden configurarse para hacer uso de estas herramientas de procesado paralelo y distribuido, tanto para acelerar el proceso, como para la resolución de problemas a gran escala, conocido como *Big Data*, en inglés, el cual es el término más utilizado en el área.

Como parte de la implementación en MATLAB se construyó una interfaz gráfica compuesta por una ventana y componentes de control como menús, botones, listas, botones de radio, cajas de texto, ventanas emergentes, entre otros.

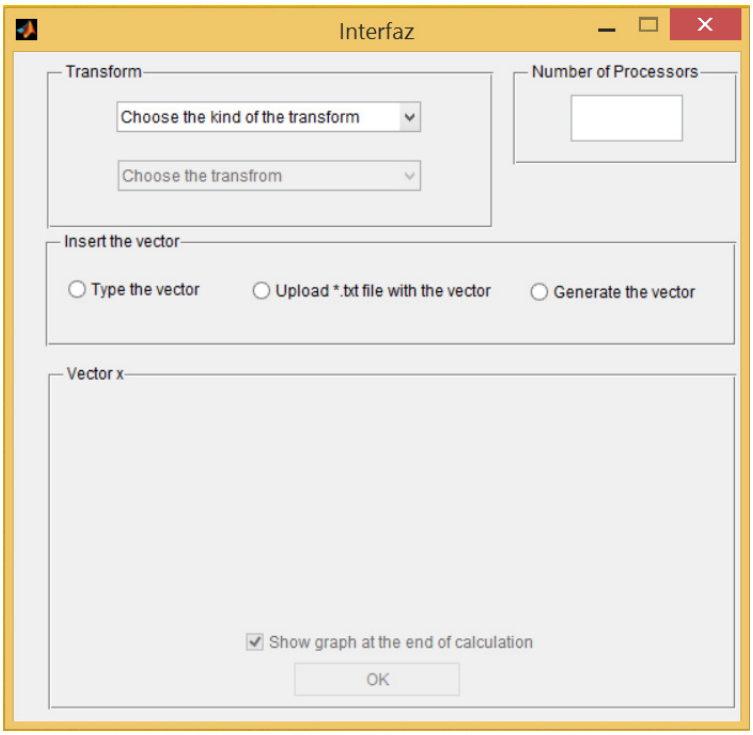

Figura 1: Vista general de la interfaz gráfica

Dicha interfaz se elaboró por medio de GUIDE, toolbox de MATLAB que crea una asociación entre un archivo *.m con código MATLAB, que contiene las funciones para iniciar la ejecución de la interfaz, los *callbacks* y la programación de las funciones que definen las propiedades, el comportamiento y los controles de cada componente, y un archivo *.fig, que contiene la distribución gráfica de cada componente de la interfaz.

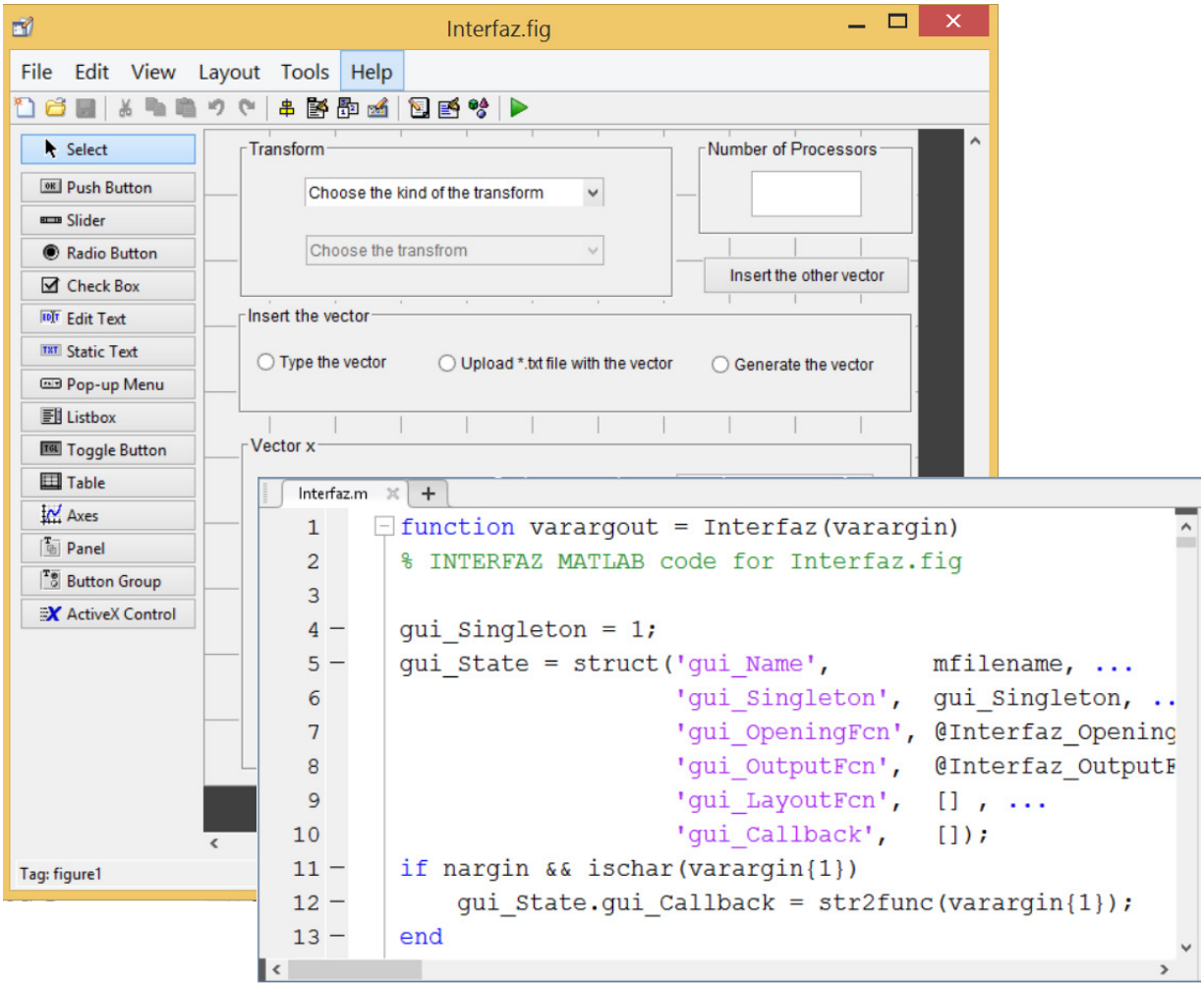

Figura 2: Archivo *.m vrs archivo *.fig

6. Resultados

Los resultados obtenidos en el presente proyecto se dividen dos partes: *resultados teóricos* y *resultados numéricos*.

- Los resultados teóricos están relacionados con el desarrollo de las formulaciones matemáticas utilizando el álgebra matricial de señales para cada una de las transormadas estudiadas. Además, se presenta un diagrama que muestra la implementación gráfica de dicha formulación y el pseudocódigo respectivo.
- Los resultados numéricos involucran la implementación en MATLAB de cada una las transformadas en el ADF, analizando el rendimiento computaciónal usando computo en paralelo. Dicho análisis se realizará calculando los tiempos de ejecución, el aceleración y eficiencia de los algoritmos, usando varias señales de prueba.

6.1. Resultados teóricos

A continuación, se presentan los resultados teóricos obtenidos. Las demostraciones respectivas están en la Sección 10: Apéndice.

6.1.1. TDF para cuaterniones y valores vectoriales

Las implementación en paralelo de TDF-1D y TDF-2D definida en (1) y (2) para $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{R})$ y $\mathbf{X} \in l^2(\mathbb{Z}_m \times \mathbb{Z}_n, \mathbb{R})$, respectivamente, se puede extender para la TDF de cuaterniones y la TDF de valores vectoriales.

- **Transformada discreta de Fourier de 1 dimensión**: Sea $\mathcal{F} = \mathbf{F}_n\mathbf{x}$ la representación matricial de la TDF-1D de $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{S})$, donde $\mathbf{F}_n \in \mathbb{S}^{n\times n}$ tal que $F_n(j,k) = \frac{1}{\sqrt{n}}\omega_n^{-jk}$, donde ω_n se definió en la Sección 3.3, dependiendo de la escogencia de $\mathbb{S}$ (ya sea $\mathbb{C}$, $\mathbb{H}$ o $\mathbb{C}^d$). Si $n = rs$, entonces $\mathbf{F}_n$ puede expresarse de la siguiente forma:

$$\mathbf{F}_n\mathbf{x} = \mathbf{P}_s(\mathbf{I}_r \otimes \mathbf{F}_s)\mathbf{P}_r\mathbf{T}_r(\mathbf{I}_s \otimes \mathbf{F}_r)\mathbf{P}_s\mathbf{x}, \tag{3}$$

  donde

$$\mathbf{T}_r = \bigoplus_{j\in\mathbb{Z}_s} \mathbf{D}_r^j, \text{ donde } \mathbf{D}_r = \bigoplus_{k\in\mathbb{Z}_r} \omega_n^{-k}$$

  y

  - si $\mathbb{S} = \mathbb{C}$ y $\mathbb{S} = \mathbb{H}$, entonces $\mathbf{P}_s = \mathbf{L}_s^n$ y $\mathbf{P}_r = \mathbf{L}_r^n$.
  - si $\mathbb{S} = \mathbb{C}^d$, entonces $\mathbf{P}_s = \mathbf{L}_s^n \otimes \mathbf{I}_d$ y $\mathbf{P}_r = \mathbf{L}_r^n \otimes \mathbf{I}_d$.

  La Figura 3 representa un diagrama del cómputo de la TDF-1D y el Algorimo 1 muestra el pseudocódigo para el cómputo de dicha transformada.

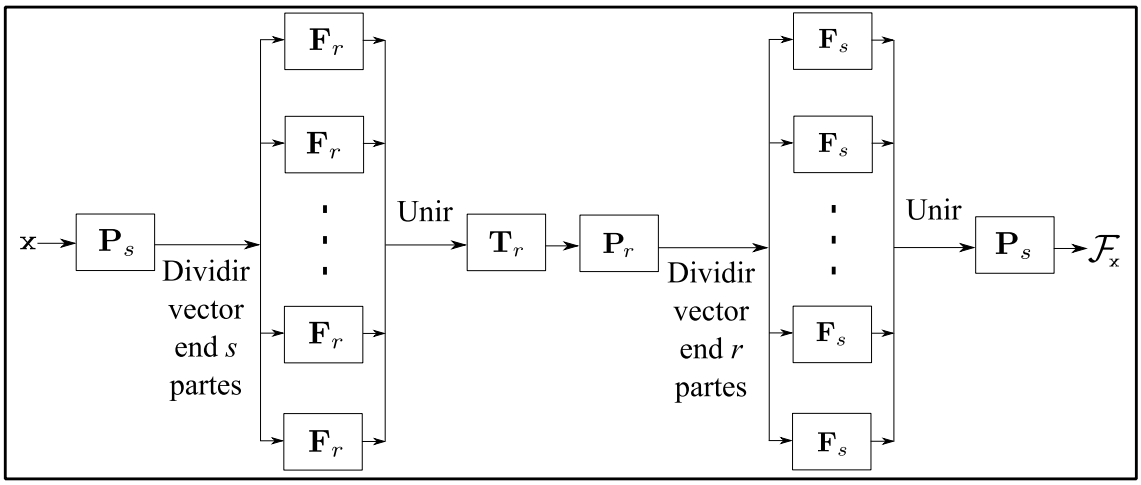

Figura 3: Diagrama del computo de la TDF-1D para una señal $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{S})$, donde $n = rs$.

Algoritmo 1: TDF-1D en paralelo

Entrada: $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{S})$, donde $n = rs$.
Salida: $\mathbf{y} \in l^2(\mathbb{Z}_n, \mathbb{S})$.

1. $\mathbf{y} \leftarrow \mathbf{P}_s\mathbf{x}$
2. $\mathbf{A} \leftarrow \mathcal{R}_{r,s}\{\mathbf{y}\}$
3. **Para** $m \leftarrow 0 : s-1$
4. $\quad \mathbf{A}(:,m) \leftarrow \mathbf{F}_r\mathbf{A}(:,m)$
5. **Fin**
6. **Para** $m \leftarrow 0 : s-1$
7. $\quad \mathbf{A}(:,m) \leftarrow \mathbf{D}_r^m\mathbf{A}(:,m)$
8. **Fin**
9. $\mathbf{y} \leftarrow \mathcal{V}\{\mathbf{A}\}$
10. $\mathbf{y} \leftarrow \mathbf{P}_r\mathbf{y}$
11. $\mathbf{A} \leftarrow \mathcal{R}_{s,r}\{\mathbf{y}\}$
12. **Para** $m \leftarrow 0 : r-1$
13. $\quad \mathbf{A}(:,m) \leftarrow \mathbf{F}_s\mathbf{A}(:,m)$
14. **Fin**
15. $\mathbf{y} \leftarrow \mathcal{V}\{\mathbf{A}\}$
16. $\mathbf{y} \leftarrow \mathbf{P}_s\mathbf{y}$

- **Transformada discreta de Fourier de 2 dimensión**: Sea $\mathcal{F}_\mathbf{X} \in \mathbb{S}^{n\times n}$ la representación matricial de la TDF-2D de $\mathbf{X} \in l^2(\mathbb{Z}_m \times \mathbb{Z}_n, \mathbb{S})$, tal que $\mathbf{F}_\mathbf{X}(m,k) = \mathfrak{F}(m,k)$, donde ω_n se definió en la Sección 3.3, dependiendo de la escogencia de $\mathbb{S}$ ($\mathbb{C}$, $\mathbb{H}$ o $\mathbb{C}^d$). $\mathbf{F}_\mathbf{X}$ puede expresarse de la siguiente forma:

$$\mathbf{F}_\mathbf{X} = \mathcal{R}_{m,n}\left\{\mathbf{P}_n(\mathbf{I}_m \otimes \mathbf{F}_n)\mathbf{P}_m(\mathbf{I}_n \otimes \mathbf{F}_m)\mathcal{V}\{\mathbf{X}\}\right\}. \tag{4}$$

donde

- si $\mathbb{S} = \mathbb{C}$ y $\mathbb{S} = \mathbb{H}$, entonces $\mathbf{P}_m = \mathbf{L}_m^{mn}$ y $\mathbf{P}_n = \mathbf{L}_n^{mn}$.
- si $\mathbb{S} = \mathbb{C}^d$, entonces $\mathbf{P}_m = \mathbf{L}_m^{mn} \otimes \mathbf{I}_d$ y $\mathbf{P}_n = \mathbf{L}_n^{mn} \otimes \mathbf{I}_d$.

La Figura 4 representa un diagrama del computo de la TDF-1D y el Algorimo 2 muestra el pseudocódigo para el cómputo de dicha transformada.

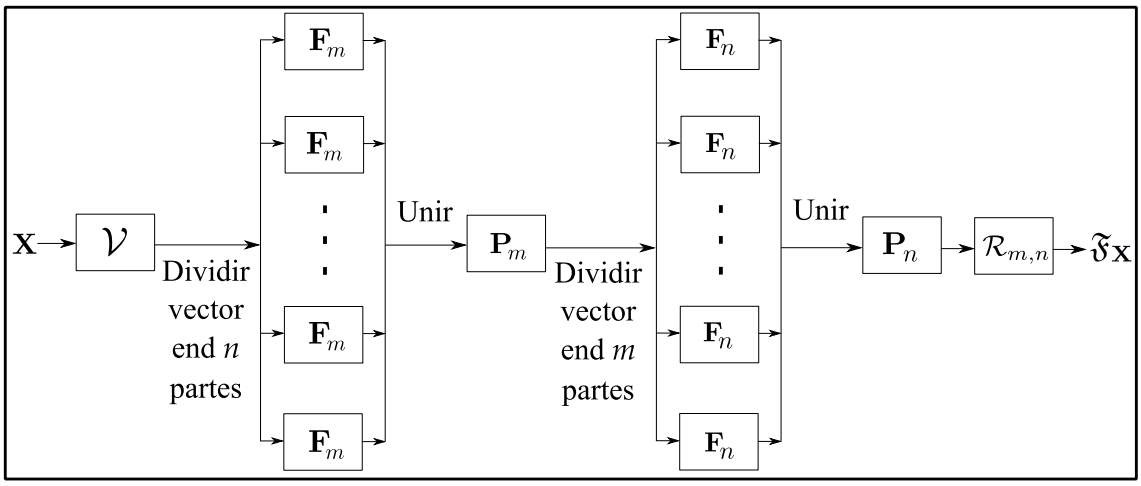

Figura 4: Diagrama del computo de la TDF-2D para una señal $\mathbf{X} \in l^2(\mathbb{Z}_m \times \mathbb{Z}_n, \mathbb{S})$.

Algoritmo 2: TDF-2D en paralelo

Entrada: $\mathbf{X} \in l^2(\mathbb{Z}_m \times \mathbb{Z}_n, \mathbb{S})$.
Salida: $\mathbf{Y} \in l^2(\mathbb{Z}_m \times \mathbb{Z}_n, \mathbb{S})$.

1. **Para** $j \leftarrow 0 : n-1$
2. $\quad \mathbf{Y}(:, j) \leftarrow \mathbf{F}_m \mathbf{X}(:, j)$
3. **Fin**
4. $\mathbf{y} \leftarrow \mathcal{V}\{\mathbf{Y}\}$
5. $\mathbf{y} \leftarrow \mathbf{P}_m \mathbf{y}$
6. $\mathbf{Y} \leftarrow \mathcal{R}_{n,m}\{\mathbf{y}\}$
5. **Para** $k \leftarrow 0 : m-1$
6. $\quad \mathbf{Y}(:, k) \leftarrow \mathbf{F}_n \mathbf{Y}(:, k)$
7. **Fin**
8. $\mathbf{y} \leftarrow \mathcal{V}\{\mathbf{Y}\}$
9. $\mathbf{y} \leftarrow \mathbf{P}_n \mathbf{y}$
10. $\mathbf{Y} \leftarrow \mathcal{R}_{m,n}\{\mathbf{y}\}$

6.1.2. Transformadas discretas en tiempo-discreto y frecuencia-discreta

La implementación en paralelo de las transformadas en tiempo-discreto y frecuencia-discreta se dividen en dos partes, según el tipo de transformada. Como se explicó en la Sección 3.4, este tipo de transformadas se clasifican en dos categorías. Sea $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{R})$:

- **Categoría 1**: Sea $\mathbf{T}_\mathbf{x} \in \mathbb{C}^{n \times n}$ la representación matricial de la transformada en TD-FD de categoría 1 de $\mathbf{x}$, tal que $\mathbf{T}_\mathbf{x}(m, k) = \mathcal{T}_\S(m, k)$, donde $\mathcal{T}_\mathbf{x}$ se definió en la Sección 3.4. Entonces $\mathbf{T}_\mathbf{x}$ puede expresarse de la siguiente forma

$$\mathbf{T}_\mathbf{x} = \frac{1}{\sqrt{n}} \mathbf{A} \odot \mathcal{R}_{n,n} \left\{ \mathbf{L}_n^{n^2} \left(\mathbf{I}_n \otimes \mathbf{F}_n \right) \left(\mathbf{h} \odot \left(\mathbf{1}_n \otimes \mathbf{x} \right) \right) \right\}, \tag{5}$$

donde $h \in \mathbb{C}^{n^2}$ tal que

$$\mathbf{h} = \begin{pmatrix} \mathbf{H}(0,:)^T \\ \vdots \\ \mathbf{H}(n-1,:)^T \end{pmatrix}.$$

Los valores de $\mathbf{A}$ y $\mathbf{H}$ se definieron en la Tabla 1. La Figura 5 representa un diagrama del computo de la transformada en TD-FD de categoría 1 y el Algorimo 3 muestra el pseudocódigo para el cómputo de dicha transformada. Para la implementación del algoritmo 3, se utiliza la siguiente propiedad: Sean $\mathbf{z} \in l^2(\mathbb{Z}_n, \mathbb{R})$, entonces $(\mathcal{R}_{n,n}\{\mathbf{z}\})^T = \mathcal{R}_{n,n}\{L_n^{n^2}\mathbf{z}\}$.

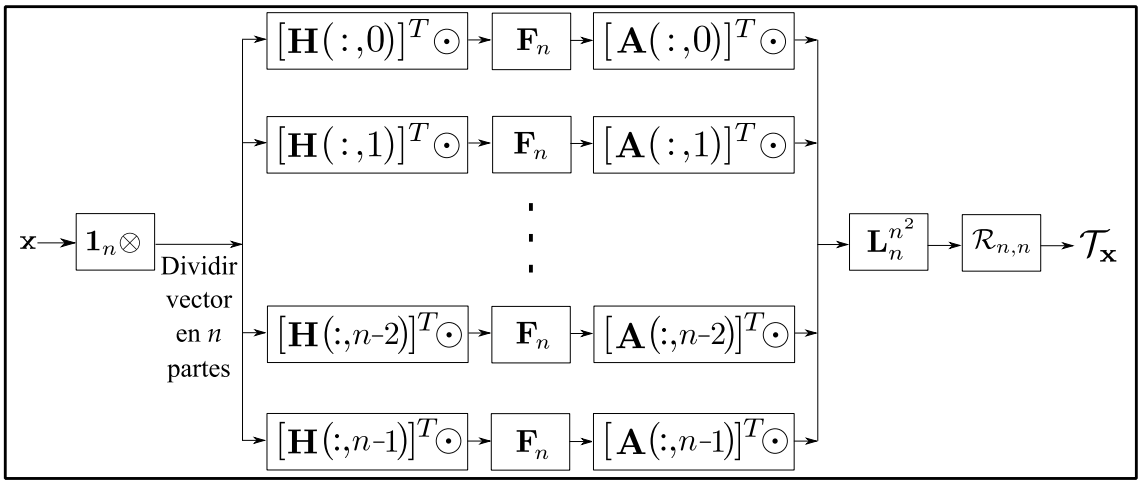

Figura 5: Diagrama del computo de la transformada en TD-FD categoría 1 para una señal $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{R})$.

Algoritmo 3: Transformada en TD-FD categoría 1 en paralelo

Entrada: $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{R})$, $\mathbf{H} \in l^2(\mathbb{Z}_n \times \mathbb{Z}_n, \mathbb{R})$ y $\mathbf{A} \in l^2(\mathbb{Z}_n \times \mathbb{Z}_n, \mathbb{R})$.
Salida: $\mathbf{T} \in l^2(\mathbb{Z}_n \times \mathbb{Z}_n, \mathbb{R})$.

1. **for** $j \leftarrow 0 : n-1$
2. $\quad \mathbf{h} \leftarrow [\mathbf{H}(j,:)]^T$
3. $\quad \mathbf{y} \leftarrow \mathbf{x} \odot \mathbf{h}$
4. $\quad \mathbf{y} \leftarrow \mathbf{F}_n\mathbf{y}$
5. $\quad \mathbf{T}(:,j) \leftarrow [\mathbf{A}(j,:)]^T \odot \mathbf{y}$
6. **end for**
7. $\mathbf{T} \leftarrow \mathbf{T}^T$

- **Categoría 2**: Sea $\mathbf{C}_{\mathbf{x},\mathbf{y}} \in \mathbb{C}^{n\times n}$ la transformada en TD-FD de categoría 2 de $\mathbf{x}$ (TDC) tal que $\mathbf{C}_{\mathbf{x},\mathbf{y}}(m,k) = \mathcal{C}_{\mathbf{x},\mathbf{y}}(m,k)$, donde $\mathcal{C}_{\mathbf{x},\mathbf{y}}$ se definió en la Sección 3.4. Entonces $\mathbf{C}_{\mathbf{x},\mathbf{y}}$ puede expresarse de la siguiente forma:

$$\mathbf{C}_{\mathbf{x},\mathbf{y}} = \mathcal{R}_{n,n}\left\{(\mathbf{I}_n \otimes \mathbf{F}_n)\mathbf{L}_n^{n^2}(\mathbf{I}_n \otimes \mathbf{F}_n)\mathcal{V}\{(\mathbf{A}_{\mathbf{x},\mathbf{y}} \odot \Phi)\}\right\}, \tag{6}$$

donde $\mathbf{A}_{\mathbf{x},\mathbf{y}} \in \mathbb{C}^{n\times n}$ es la función discreta de ambigüedad (FDA) de $\mathbf{x},\mathbf{y}$.

La Figura 6 representa un diagrama del computo de la transformada en TD-FD de categoría 2 y el Algorimo 4 muestra el pseudocódigo para el cómputo de dicha transformada. Para la implementación del algoritmo 4, se utiliza las siguientes propiedades: Sean $\mathbf{z} \in l^2(\mathbb{Z}_n,\mathbb{R})$ y $\mathbf{R} \in l^2(\mathbb{Z}_n \times \mathbb{Z}_n,\mathbb{R})$, entonces $(\mathcal{R}_{n,n}\{\mathbf{z}\})^T = \mathcal{R}_{n,n}\{L_n^{n^2}\mathbf{z}\}$ y $\mathcal{V}\{\mathbf{R}^T\} = L_n^{n^2}\mathcal{V}\{\mathbf{R}\}$.

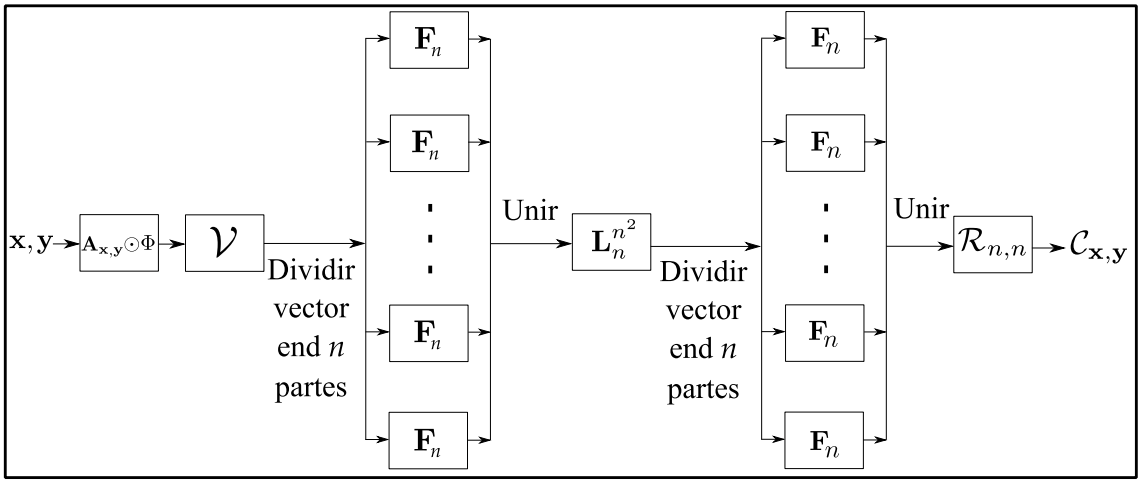

Figura 6: Diagrama del computo de la transformada en TD-FD categoría 1 para una señal $\mathbf{x} \in l^2(\mathbb{Z}_n,\mathbb{R})$.

Algoritmo 4: Transformada en TD-FD categoría 2 en paralelo

Entrada: $\mathbf{x} \in l^2(\mathbb{Z}_n,\mathbb{R})$, $\mathbf{y} \in l^2(\mathbb{Z}_n,\mathbb{R})$ y $\Phi \in l^2(\mathbb{Z}_n \times \mathbb{Z}_n,\mathbb{R})$.
Salida: $\mathbf{C} \in l^2(\mathbb{Z}_n \times \mathbb{Z}_n,\mathbb{R})$.

1. $\mathbf{A} \leftarrow$ FDA de $\mathbf{x}$ y $\mathbf{y}$ (señal eco)
2. $\mathbf{C} \leftarrow \mathbf{A} \odot \Phi$
3. **for** $j \leftarrow 0 : n-1$
4. $\quad \mathbf{C}(:,j) = \mathbf{F}_n\mathbf{C}(:,j)$
5. **end for**
6. $\mathbf{C} \leftarrow \mathbf{C}^T$
7. **for** $k \leftarrow 0 : n-1$
8. $\quad \mathbf{C}(:,k) = \mathbf{F}_n\mathbf{C}(:,k)$
9. **end for**

6.2. Resultados numéricos

6.2.1. Información General

La presente investigación se realizó utilizando una computadora con procesadores multinúcleos. Dicha computadora consiste de 4 núcleso con procesador Intel® Core i7-3632QM CPU, con un reloj del sistema de 2.20 GHz y 8 GB de RAM.

La implementación de todos las transformadas discretas se desarrolló usando MATLAB y la herramienta computacional de computo en paralelo (*Parallel Computing Toolbox*, en inglés). MATLAB utiliza dos tipos de paralelismo: múltiples hilos integrados y *workers*. En esta investigación se utiliza *workers*. Este enfoque permite un mayor control sobre los algoritmos en paralelo que con una función de múltiples hilos. Para eso, se utilizan construcciones de programación tales como paralelo con bucles, en especial el comando `parfor`.

6.2.2. Resultados y discusiones

En esta sección, se mostrarán los tiempos de ejecución de cada una de las transformadas presentadas en este informe. Dichos tiempos de ejecución representan la implementación en paralelo y la implementación en forma secuencial, es decir, sin nunguna implementación en paralelo. La implementación en paralelo se realizó usando p procesadores, donde $p = 1, 2, 3$ y 4, las cuales fueron presentadas en la Sección 6.1. La implementación secuencial se realizará utilizando la fórmula original de cada una de la transformadas, las cuales fueron presentadas en la Sección 3.3. La finalidad de dichas implementaciones es de corroborar que las implementaciones en paralelo minimizan el tiempo de ejecución.

Como se desarrolló en la Sección 6.1, las transformadas presentadas en este informe se clasifican en 4 grupos: TDF-1D, TDF-2D y las trasformadas en TD-FD de categoría 1 y 2. Se presentan los resultados numéricos de cada una de ellas, para un un caso específico y para un tipo de señal específico. Dichas escogencias son:

1. TDF-1D de señales complejas:
   - Tipo de señal: Single Chirp Signal (Type 1).
   - Dimensiones: $n = 2^{13}, 2^{14}, 2^{15}$.
2. TDF-1D de señales de valores vectoriales:
   - Tipo de señal: Wiener Vector-Valued Signal.
   - Dimensiones: $n = 2^{12}, 2^{13}, 2^{14}$ y $d = 5$.

3. TDF-2D de señales de quaterniones:
   - Tipo de señal: Imagen Lena.
   - Dimensiones: $n \times n = 2^6 \times 2^6, 2^7 \times 2^7, 2^8 \times 2^8$.
4. Trasformadas en TD-FD de categoría 1:
   - Tipo de tansformada: Transformada discreta de Fourier en tiempo corto.
   - Tipo de señal: Single Chirp Signal (Type 1) y Hamming window.
   - Dimensiones: $n = 2^{11}, 2^{12}, 2^{13}$.
5. Trasformadas en TD-FD de categoría 2:
   - Tipo de tansformada: Wigner.
   - Tipo de señal: Single Chirp Signal (Type 1).
   - Dimensiones: $2^{11}, 2^{12}, 2^{13}$.

Las Tablas 3, 5, 4, 6 y 7 muestran los tiempos de ejecución, en segundos (s), de la implementación en paralelo y secuencial. En cada tabla se muestra la ventaja del uso de procesadores multinúcleos y un ambiente en paralelo para minimizar el alto tiempo de ejecución de cada una de las transformadas. Lo anterior se debe a que el cómputo en paralelo es una forma de cálculo en la que muchos cálculos se llevan a cabo simultáneamente [1, 37], que opera en el principio de que los grandes problemas, en su mayoría, se pueden dividir en problemas más pequeños, que luego se resuelven simultáneamente, y minimizan el tiempo de ejecución [35, 37].

		n		
Transformada	p	8192	16384	32768
TDF-1D	*	132.7	746.3	2651.1
	1	0.938	2.639	10.61
	2	0.624	1.529	5.929
	3	0.549	1.163	4.380
	4	0.538	1.073	3.995

Cuadro 3: Tiempo de ejecución para el cómputo de la TDF-1D de señales en $l^2(\mathbb{Z}_n, \mathbb{C})$, usando implementación en paralelo con p procesadores. * representa el tiempo de ejecución de la implementación en forma secuencial.

		n		
Transformada	p	4096	8192	16384
TDF-1D	*	853.5	13408	+15000
	1	18.78	106.7	263.4
	2	17.25	80.44	180.9
	3	12.75	57.35	154.7
	4	6.481	32.67	82.65

Cuadro 4: Tiempo de ejecución para el cómputo de la TDF-1D de señales en $l^2(\mathbb{Z}_n, \mathbb{C}^d)$, usando implementación en paralelo con p procesadores. * representa el tiempo de ejecución de la implementación en forma secuencial.

		$n \times n$		
Transformada	p	64×64	128×128	256×256
TDF-2D	*	101.5	253.2	489.1
	1	1.642	5.943	56.91
	2	1.639	5.922	56.80
	3	1.632	5.915	56.67
	4	1.627	5.898	56.54

Cuadro 5: Tiempo de ejecución para el cómputo de la TDF-2D de señales en $l^2(\mathbb{Z}_n \times \mathbb{Z}_n, \mathbb{H})$, usando implementación en paralelo con p procesadores. * representa el tiempo de ejecución de la implementación en forma secuencial.

		n		
Transformada	p	2048	4096	8192
TD-FD Categoría 1	*	131.15	1012.14	2931.20
	1	5.980	27.207	170.943
	2	2.747	14.060	78.207
	3	1.211	8.172	32.331
	4	0.926	5.616	24.942

Cuadro 6: Tiempo de ejecución para el cómputo de la transformada TD-FD de categoría 1, de señales en $l^2(\mathbb{Z}_n, \mathbb{C})$, usando implementación en paralelo con p procesadores. * representa el tiempo de ejecución de la implementación en forma secuencial.

Transformada	p	n		
		2048	4096	8192
TD-FD Categoría 2	*	199.745	2041.17	4510.52
	1	17.674	96.004	552.684
	2	14.078	68.023	325.784
	3	13.251	59.182	262.333
	4	13.087	56.575	238.341

Cuadro 7: Tiempo de ejecución para el cómputo de la transformada TD-FD de categoría 2, de señales en $l^2(\mathbb{Z}_n, \mathbb{C})$, usando implementación en paralelo con p procesadores. * representa el tiempo de ejecución de la implementación en forma secuencial.

6.3. Resultados publicados

En [32] se muestran algunos resultados numéricos para las transformadas: función discreta de ambiguedad (FDA), la transformada discreta de Zak (TDZ), la transformada discreta de Fourier en tiempo corto (TDFTC), la transformada discreta chirp-Fourier (TDCF) y sus derivadas: la transformada discreta chirp-Fourier modificada (TDCFM) y nueva la transformada discreta chirp-Fourier (NTDCF). En este artículo se muestra como el álgebra matricial de señales contribuye al análisis, diseño e implementación de algoritmos en paralelo en procesadores multinúcleo. Se concluye que hay ventaja en usar procesadores multinúcleo y un entorno de computación en paralelo para minimizar los tiempos de ejecución. También, se muestra como la aceleración (definida como la razón entre los tiempos de ejecución de las implementaciones secuenciales y en paralelo, y es un valor típicamente entre 1 y el número de procesadores) y eficiencia (definida como la razón entre la aceleración y el número de procesadores) mejora cuando el número de procesadores lógicos y la longitud de la señal aumentam teniendo un aumento de acelaración superlineal.

Aunque la interfaz no se agregó la transformada discreta de Fourier de valores vectoriales (TDFVV), en [33] se hace un estudio de dicha transformada. En particular, se presentan las condiciones necesarias y suficientes para que sea invertible. Además, en [34] se muestran los resultados teóricos de su representación matricial por bloques y su implementación en paralelo. Se muestra como el uso de procesadores multinúcleo y un entorno de computación en paralelo minimizan el tiempo de ejecución. Se hace un análisis de la aceleración y la eficiencia, obteniendo que la aceleración aumenta cuando el número de procesadores aumenta independientemente de la logitud del vector. Se obtiene una buena eficiencia cuando el número de procesadores es 2, pero la eficiencia disminuye cuando el número de procesadores aumenta independientemente de la logitud del vector.

6.4. Interfaz gráfica

Está interfaz le permitirá al usuario trabajar con las distintas transformadas de Fourier implementadas en paralelo.

6.4.1. Organización de la interfaz gráfica

Se clasificaron las transformadas según las categorías DFT for complex signals, DFT for quaternions signals y Time-Frecuency Transforms.

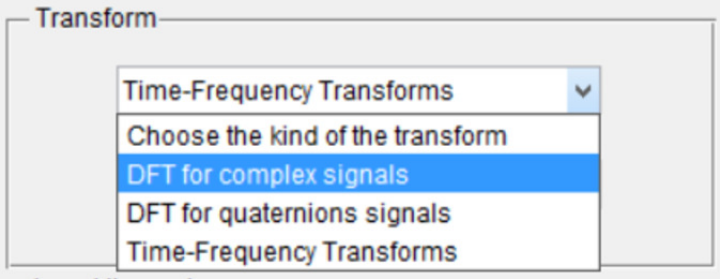

Figura 7: Menú para elegir el tipo de transformada

Cada categoría presenta un conjunto de DFT, según corresponda:

1. DFT for complex signals
    - DFT 1D complex parallel
    - DFT 2D complex parallel

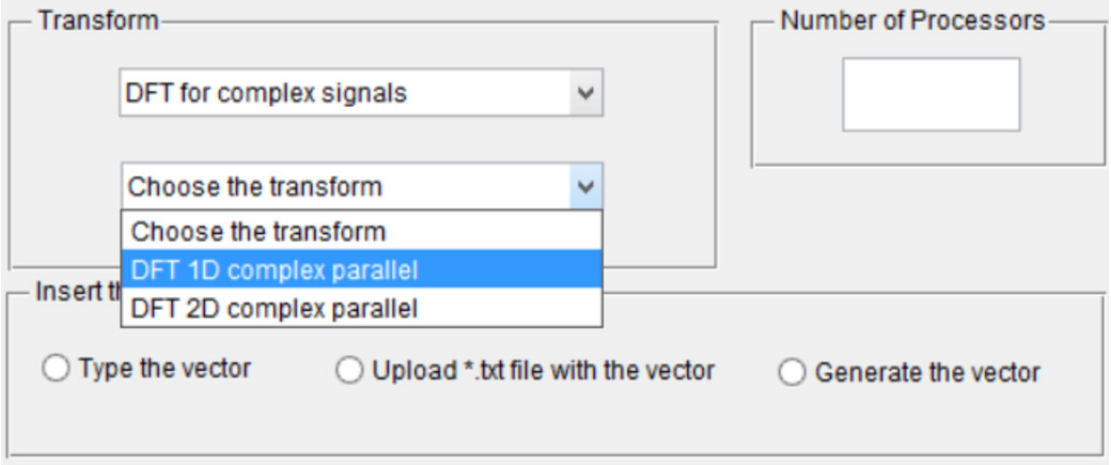

Figura 8: Categoría DFT for complex signals

2. DFT for quaternions signals

- QDFT 1D
- QDFT 2D

3. Time-Frecuency Transforms

- DAF
- DZT
- MDCFT
- Cohen
- DSTFT
- DCFT
- NDCFT

Cada transformada requiere de una o dos entradas que, en la mayoría de los casos, es un vector. Para ello la interfaz cuenta con tres opciones: digitar, cargar desde un archivo *.txt o generar por medio de alguna función y parámetros determinados.

Figura 9: Opciones para los datos de entrada

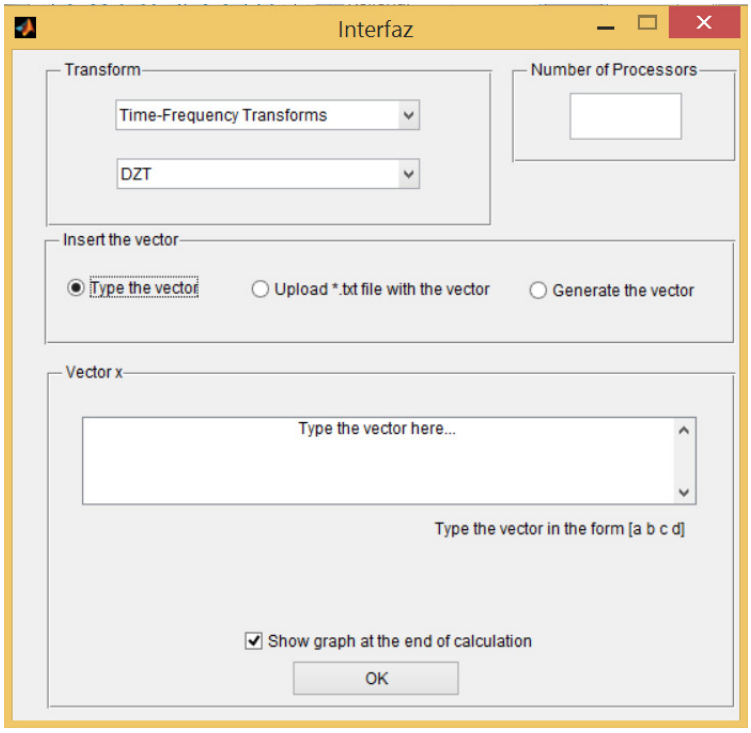

Figura 10: Opción de digitar vector

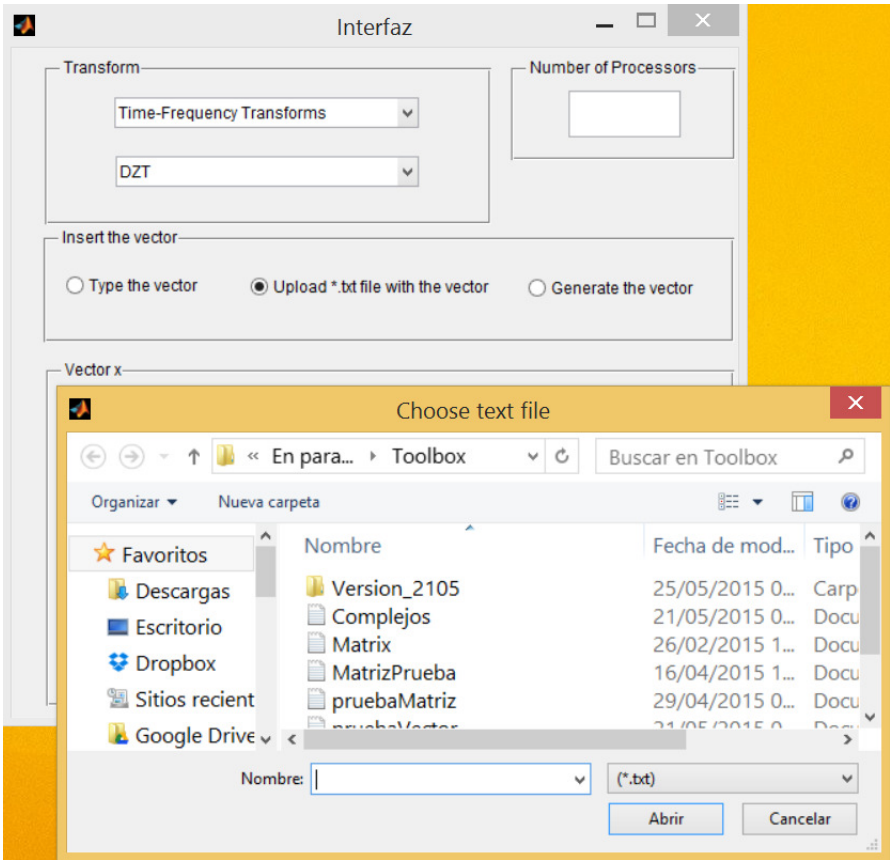

Figura 11: Opción de cargar vector desde un archivo *.txt

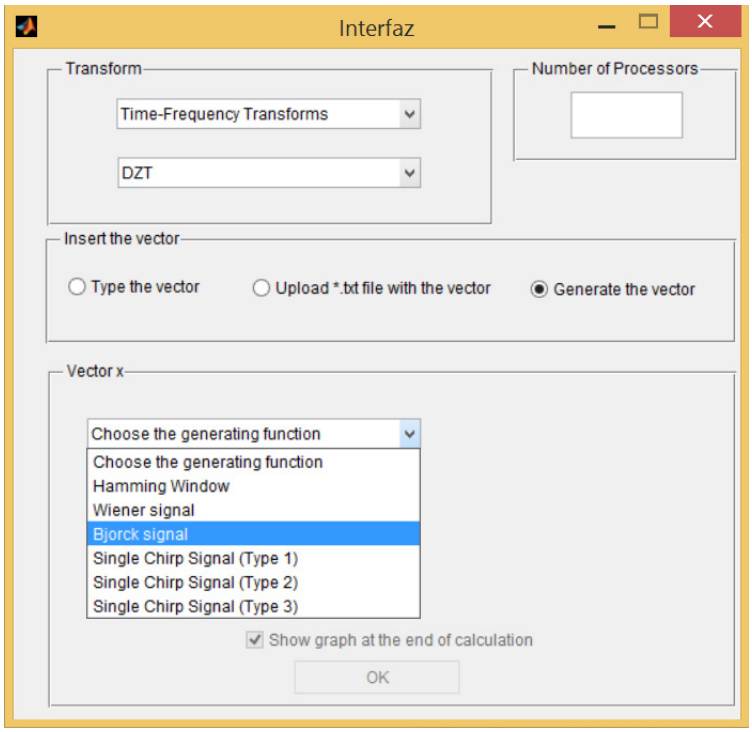

Figura 12: Opción de generar vector a partir de una de las funciones definidas

Sin embargo, hay casos particules para el ingreso de los datos:

- QDFT 1D y la DFT 2D complex parallel no cuentan con la opción “Generate” .
- En QDFT 2D la opción “Generate” cambia por “Upload imagen” , donde la imagen se transformará en una representación RGB por medio de tres matrices para construir la matriz cuaternion que utilizará la transformada.

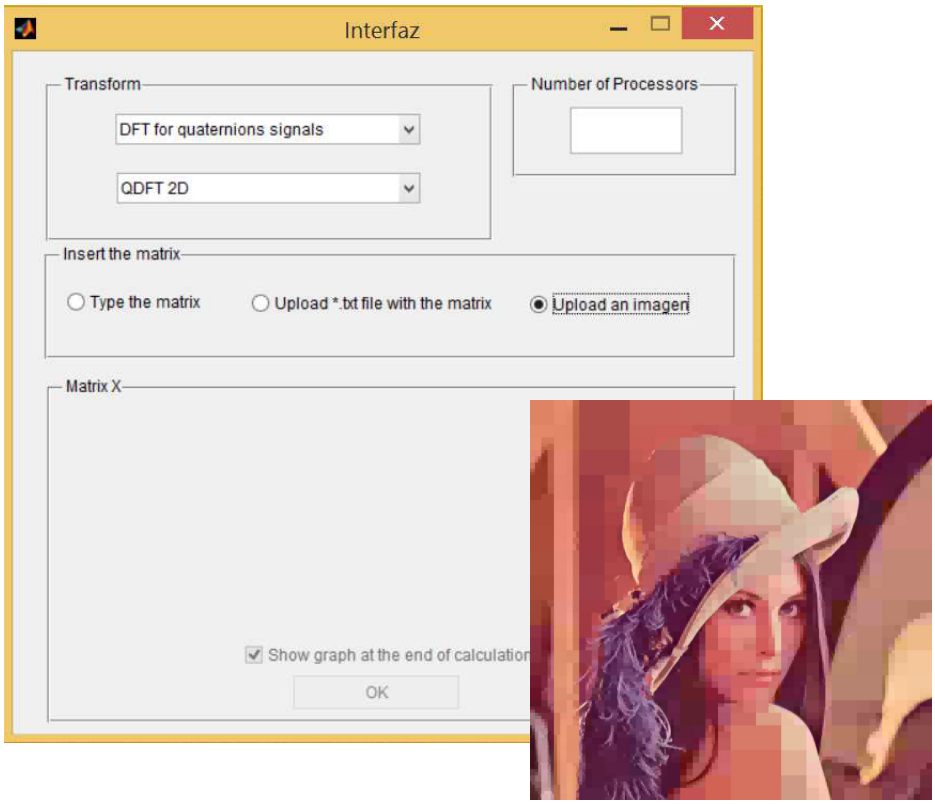

Figura 13: Cargar imagen para QDFT 2D

- Las entradas de DFT 2D complex parallel y QDFT 2D deben ser matrices en vez de vectores.

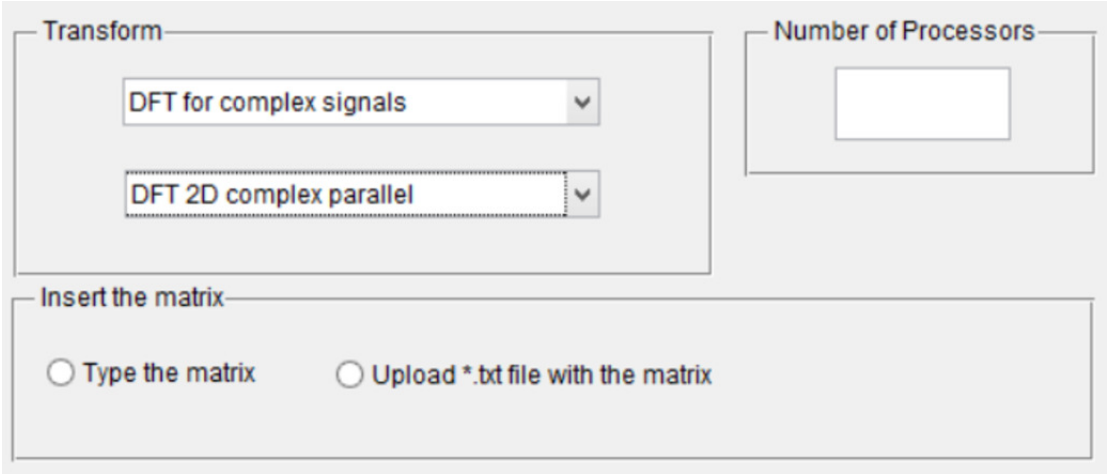

Figura 14: Opciones de ingreso para DFT 2D complex parallel

- El vector de entrada para DFT 1D complex parallel y QDFT 1D debe tener una dimensión especial, producto de dos números r y s, por lo que al seleccionar dichas transformadas

la interfaz solicitará primero el valor de esos números cuyo producto debe coindir con la dimensión del vector de entrada.

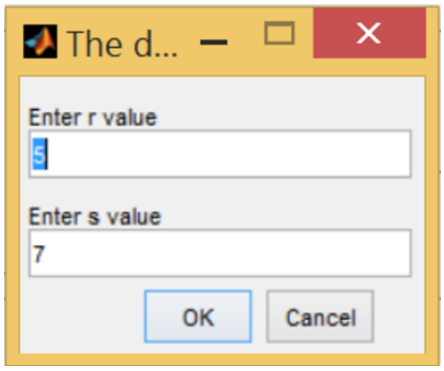

Figura 15: Ingreso de r y s

- DAF, DSTFT y Cohen requieren de dos vectores de entrada en vez de uno por lo que al seleccionar dichas transformadas aparacerá un botón extra en la parte superior derecha de la ventana. Luego de ingresar el primer vector deben presionar dicho botón y después de ingresar el segundo vector se debe presionar el botón OK, que se encuentra en la parte de abajo de la ventana, para iniciar la transformada según corresponda.

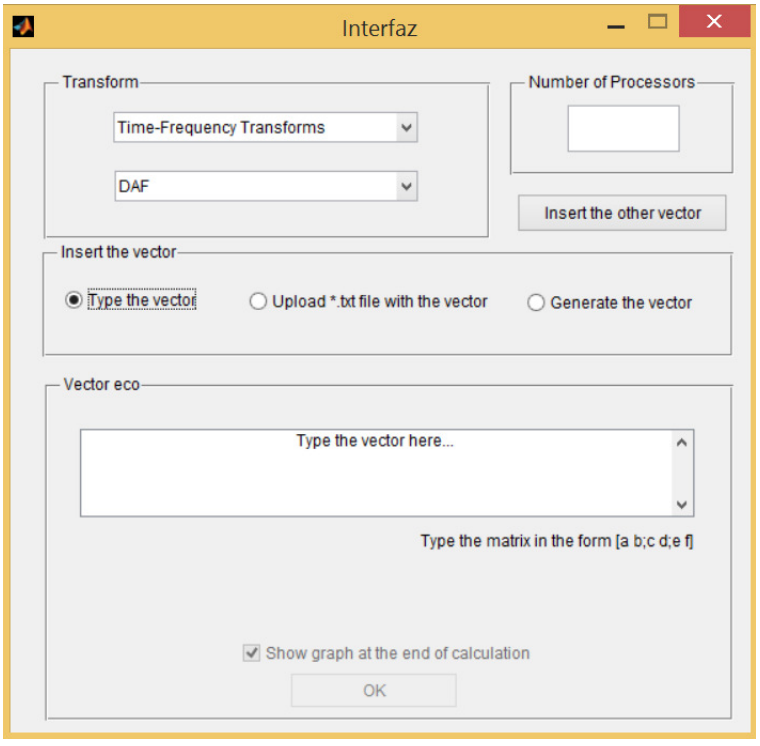

Figura 16: Interfaz para el ingreso de dos vector

- Para Cohen, además, se debe determinar un núcleo por lo que al seleccionar dicha transformada la interfaz solicitará que se indique el núcleo a utilizar.

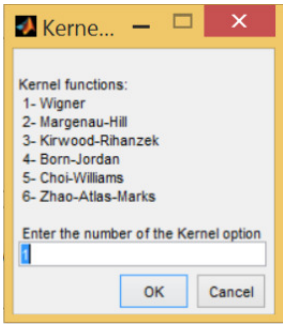

Figura 17: Selección del núcleo para Cohen

- Las transformadas que tienen habilitada la opción "Generate" cuentan con una lista de opciones para dicha generación. La lista por defecto es la siguiente:

  - Hamming Window

$$\mathbf{x}_n = a - (1-a)\cos\left(\frac{2\pi n}{N-1}\right)$$

  - Wiener Signal

$$\mathbf{x}_n = exp\left(\frac{2\pi n^2}{M}\right);\ M = \begin{cases} N; & \text{si } N \text{ es par} \\ 2N; & \text{sino} \end{cases}$$

  - Bjork Signal

$$\mathbf{x}_n = exp\left(2\pi \cdot \theta(n,p)\right);\ \theta(n,p) = \frac{k}{n} \cdot \arccos\left(\frac{1}{1+\sqrt{p}}\right)$$

  - Single Chirp Signal (Type 1)

$$\mathbf{x}_n = \omega_N^{-(ln^2+kn)};\ \omega_N = exp\left(\frac{2\pi i}{N}\right)$$

  - Single Chirp Signal (Type 2)

$$\mathbf{x}_n = \omega_N^{-(ln^2/2+kn)};\ \omega_N = exp\left(\frac{2\pi i}{N}\right)$$

  - Single Chirp Signal (Type 3)

$$\mathbf{x}_n = \omega_N^{-\left(l(n-N/2)^2/2+k(n-N/2)\right)};\ \omega_N = exp\left(\frac{2\pi i}{N}\right)$$

En todos los casos N es el tamaño del vector, n varía de 0 a $N-1$ y los demás parámetros constantes.

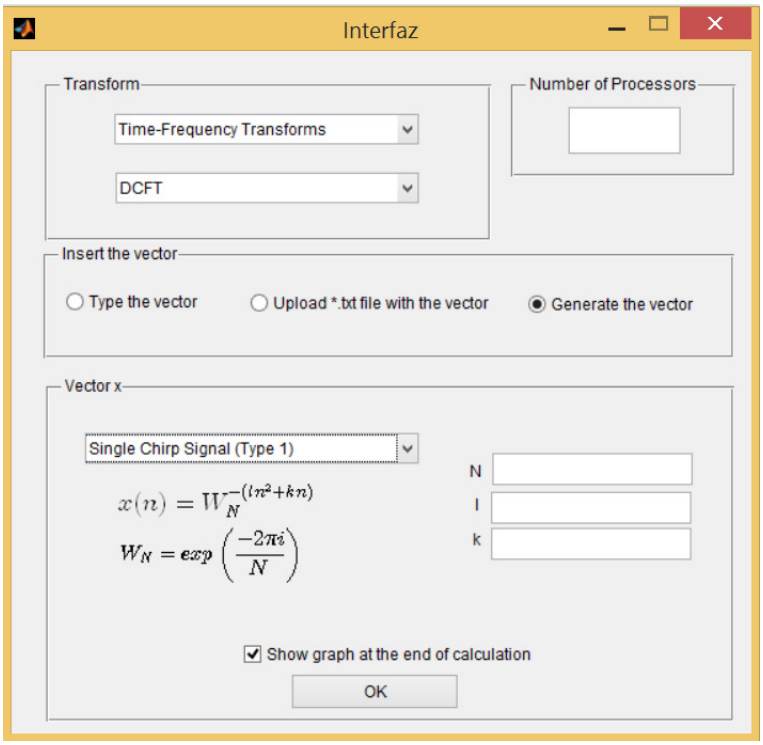

Figura 18: Generación de un vector por medio de Single Chirp Signal (Type 1)

Sin embargo, hay una excepción. El primer vector de DSTFT cuenta con la siguiente lista para la opción "Generate" , en vez de la lista por defecto indicada anteriormente:

- Rectangular

$$\mathbf{x}_n = \begin{cases} 1; & \text{si } n \in [0, T] \\ 0; & \text{sino} \end{cases}$$

- Hann

$$\mathbf{x}_n = \frac{1}{2}\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right)$$

- Hamming

$$\mathbf{x}_n = a - (1-a)\cos\left(\frac{2\pi n}{N-1}\right)$$

- Blackman

$$\mathbf{x}_n = \frac{1-a}{2} - \frac{1}{2}\cos\left(\frac{2\pi n}{N-1}\right) + \frac{a}{2}\cos\left(\frac{4\pi n}{N-1}\right)$$

- Blackman-Harris

$$\mathbf{x}_n = \sum_{i=0}^{3}(-1)^i a_i \cos\left(\frac{2i\pi n}{N-1}\right)$$

$$a_0 = 0{,}35875,\ a_1 = 0{,}48829,\ a_2 = 0{,}14128,\ a_3 = 0{,}01168$$

- Blackman-Nuttall

$$\mathbf{x}_n = \sum_{i=0}^{3} (-1)^i a_i \cos\left(\frac{2i\pi n}{N-1}\right)$$

$$a_0 = 0{,}3635819,\ a_1 = 0{,}4891775,\ a_2 = 0{,}1365995,\ a_3 = 0{,}0106411$$

- Flat top

$$\mathbf{x}_n = \sum_{i=0}^{4} (-1)^i a_i \cos\left(\frac{2i\pi n}{N-1}\right)$$

$$a_0 = 1,\ a_1 = 1{,}93,\ a_2 = 1{,}29,\ a_3 = 0{,}388,\ a_4 = 0{,}032$$

- Gauss

$$\mathbf{x}_n = exp\left(-\frac{1}{2}\left(\frac{2n-(N-1)}{s(N-1)}\right)^2\right);\ s \leq 0{,}5$$

- Triangular

$$\mathbf{x}_n = \frac{N}{2} - \left|n - \frac{N-1}{2}\right|$$

- Bartlett

$$\mathbf{x}_n = \frac{N-1}{2} - \left|n - \frac{N-1}{2}\right|$$

- Bartlett-Hann

$$\mathbf{x}_n = a_0 - a_1\left|\frac{n}{N-1} - \frac{1}{2}\right| - a_2 \cos\left(\frac{2\pi n}{N-1}\right)$$

$$a_0 = 0{,}62,\ a_1 = 0{,}48,\ a_2 = 0{,}38$$

- Kaiser

$$\mathbf{x}_n = \frac{I_0\left(\pi a\sqrt{1-(2n/(N-1)-1)^2}\right)}{I_0(\pi a)};\ I_0(z) = \sum_{k=0}^{\infty} \frac{(z^2/4)^k}{k!\Gamma(k+1)}$$

Similarmente al caso por defecto, N es el tamaño del vector, n varía de 0 a $N-1$ y los demás parámetros constantes.

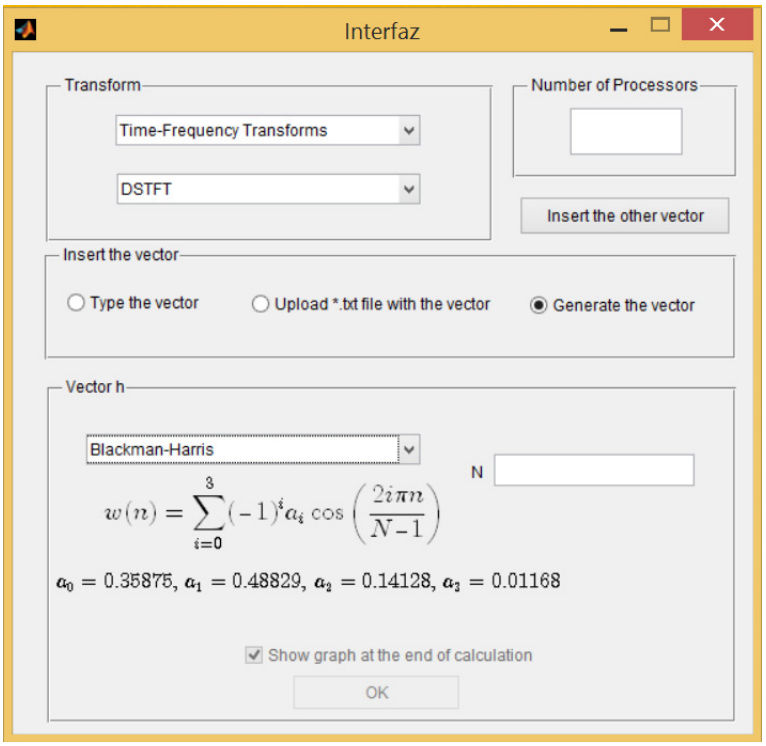

Figura 19: Generación de un vector para DSTFT por medio de Blackman-Harris

Por otro lado, en la casilla que se ubica en la esquina superior derecha de la ventana se pueden indicar la cantidad de procesadores con lo que desea trabajar en paralelo, la opción por defecto es un 1 procesador.

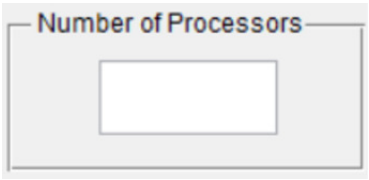

Figura 20: Cantidad de procesadores

Por medio del botón OK se ejecuta la transformada seleccionada, luego de haber insertado de manera correcta el o los vectores o matrices, según corresponda. Además, antes del botón OK hay una opción para generar la gráfica correspondiente a los resultados, esta es una casilla opcional la cual aparecerá seleccionada por defecto.

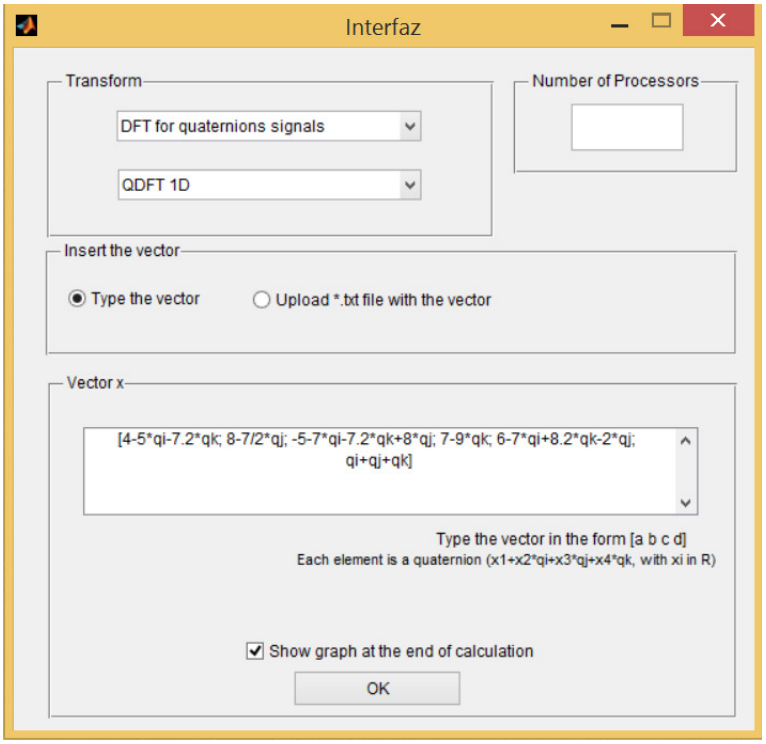

Figura 21: Botón OK listo para ejecutar la transformada

Al finalizar la ejecución de la transformada es posible guardar los resultados en un archivo *.txt, si así lo desea.

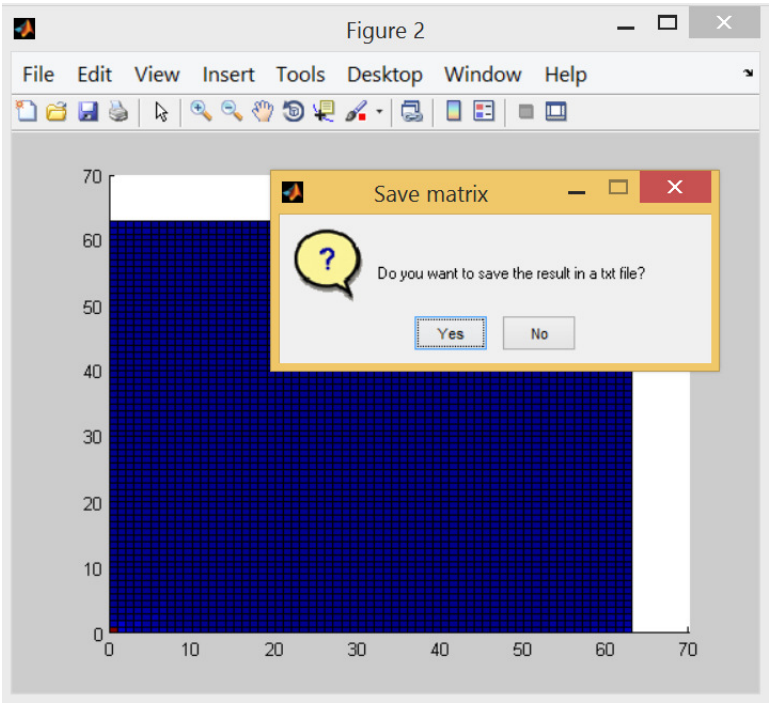

Figura 22: Posibles resultados: Gráfica y archivo *.txt

Además de las transformadas mencionadas anteriormente hay un grupo de transformadas que no se incluye en la interfaz gráfica, que son las TDF-1D y TDF-2D de valores vectoriales. Lo anterios se debe a que las aplicaciones encontradas en la literatura no involucran el cómputo de la gráfica, ya que es considerada hasta la fecha como una herramienta teórica para ciertos problemas en sonares (ver, por ejemplo, [4, 5, 29]). Pero dichas implementaciones (funciones desarroladas en MATLAB) sí se incluyen como parte del *toolbox* final.

6.5. Manual de usuario

Se diseñó un manual de usuario para el uso de dicho ambiente computacional.

En dicho manual se muestra la definición de cada una de las transformadas, su representación matricial y el comando respectivo implementado en MATLAB para su ejecución. Se explica la implementación en paralelo. Además, se desarrollaron algunos ejemplos para cada transformada.

2.1 1-D SIGNALS

The discrete Fourier transform (DFT) of $x \in L^2(\mathbb{Z}_n, \mathbb{S})$ is represented as $\mathscr{F}_x : \mathbb{Z}_n \to \mathbb{S}$ such that

$$\mathscr{F}_x(k) = \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} \omega_n^{-mk} x(m), \tag{1}$$

where ω_n is called the kernel of DFT. There are different kinds of kernels:

- if $\mathbb{S} = \mathbb{C}$, then $\omega_n = e^{i\frac{2\pi}{n}}$ and it is called discrete Fourier transform (DFT) for complex signals, or just DFT.
- if $\mathbb{S} = \mathbb{H}$, then $\omega_n = e^{\mu\frac{2\pi}{n}}$, where μ is any pure unit quaternion and it is called Quaternions DFT (QDFT).
- if $\mathbb{S} = \mathbb{C}^d$, then $\omega_n \in \mathbb{C}^{d\times d}$ and it is called Vector-Valued DFT (VV DFT).

2.1.1 PARALLEL IMPLEMENTATION

Let $\mathscr{F}_x = F_n x$ be the matrix representation of DFT of x, where $F_n \in \mathbb{S}^{n\times n}$ such that $F_n(j,k) = \frac{1}{\sqrt{n}}\omega_n^{-jk}$. If $n = rs$, then the matrix formalism can be used to express F_n as factorizations of matrices:

$$F_n x = L_s^n (I_r \otimes F_s) L_r^n T_r^n (I_s \otimes F_r) L_s^n x, \tag{2}$$

Figura 23: Definición de una de las trasformadas y su implementación en paralelo

3.1.2 FUNCTIONS OF TOOLBOX

- `p_daf(x,y,p)`: Compute discrete ambiguity function of complex vector `x` and eco vector `y` using `p` cores.
- `p_dstft(x,h,p)`: Compute discrete short-time Fourier transform of complex vector `x` and window vector `h` using `p` cores.
- `p_dzk(x,p)`: Compute discrete zak transform of complex vector `x` using `p` cores.
- `p_dcft(x,p)`: Compute discrete chirp-Fourier transform of complex vector `x` using `p` cores.
- `p_mdcft(x,p)`: Compute modified discrete chirp-Fourier transform of complex vector `x` using `p` cores.
- `p_ndcft(x,p)`: Compute new discrete chirp-Fourier transform of complex vector `x` using `p` cores.

Figura 24: Comandos en MATLAB para algunas de las transformadas

Se incluye una sección para el uso del ambiente gráfico y de la manipulación de éste. Además, se desarrollaron ejemplos para cada transformada.

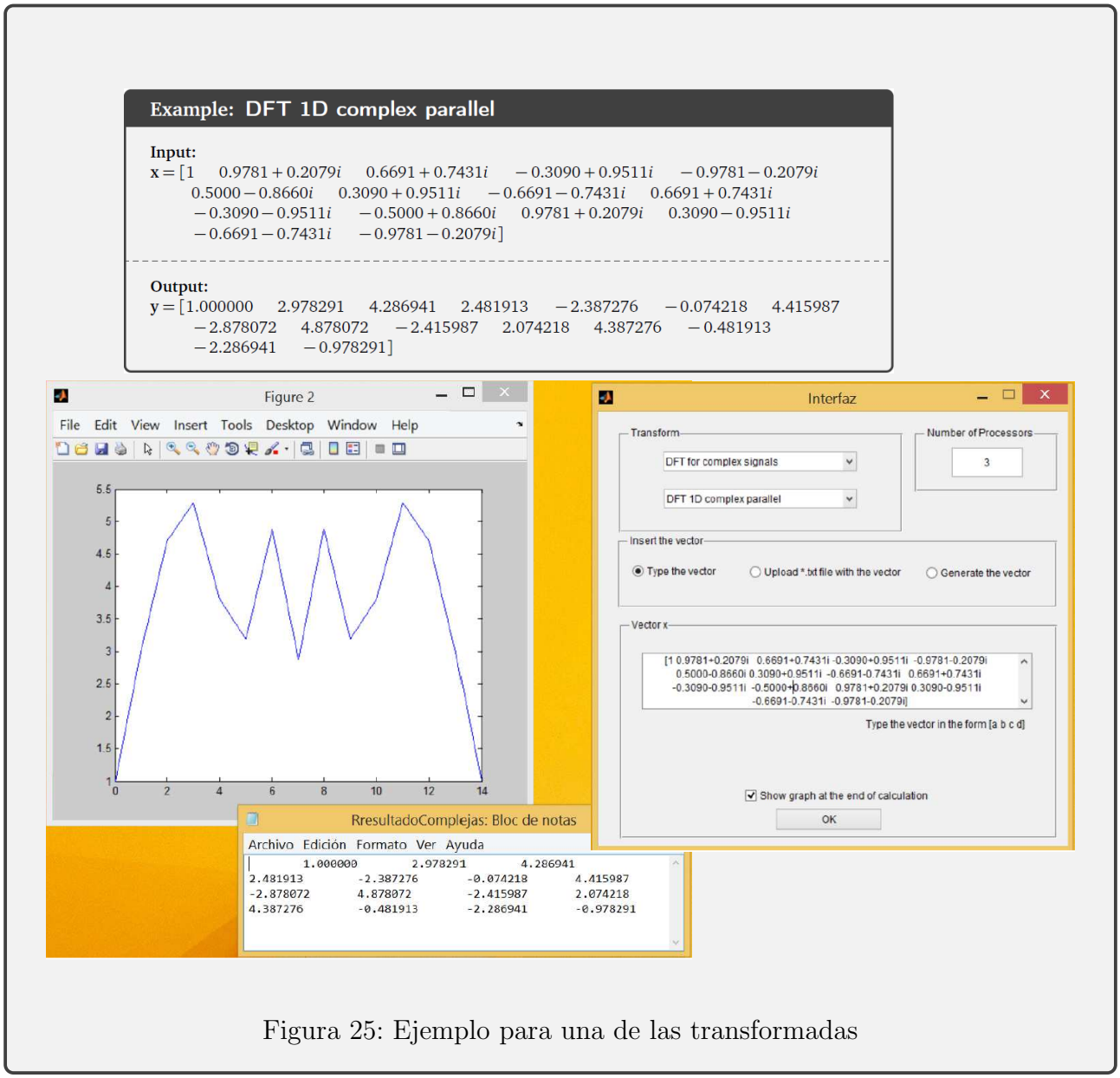

Figura 25: Ejemplo para una de las transformadas

Este manual está disponible en un documento con extensión pdf.

7. Divulgación

El proyecto de investigación ha sido difundido mediante los siguientes artículos:

- **Artículo** 1:
  - *Título*: A Mathematical Framework for Parallel Computing of Discrete-Time Discrete-Frequency Transforms in Multi-Core Processors.
  - *Volumen, número, páginas y año*: **8**, No. 6, 2795-2801 (2014)
  - *Revista Científica*: Applied Mathematics & Information Sciences (Indexada en *Science Citation Index Expanded* y *SCOPUS*, entre otros.)
  - *DOI*: 10.12785/amis/080615
- **Artículo** 2:
  - *Título*: Application of block matrix theory to obtain the inverse transform of the vector-valued DFT.
  - *Volumen, número, páginas y año*: **9**, No. 53, 2567-2577 (2015)
  - *Revista Científica*: Applied Mathematical Sciences (Indexada en *SCOPUS* y *ROAD*, entre otros.)
  - *DOI*: 10.12988/ams.2015.52125

Además, un artículo está en revisión en estos momentos. El título de dicho artículo es *A Parallel Framework with Block Matrices of a Discrete Fourier Transform for Vector-Valued Discrete-Time Signals* (Ver versión presentada a revisión adjunto a este documento).

Por último, se someterá un artículo a una revista indexada por definir. El título de dicho artículo es *A Mathematical Framework for Parallel Implementation in Multi-Core Processors of Finite Transforms in Discrete Fourier Analysis.*

8. Conclusiones

La investigación permite llegar a las siguientes conclusiones

1. Un grupo de transformadas discretas en el análisis discreto de Fourier puede ser calculada utilizando la transformada discreta de Fourier de 1 dimensión, utilizando cómputo en paralelo.

2. El cómputo en paralelo puede ser expresado utilizando un álgebra matricial de señales, el cual consiste en un ambiente matemático compuesto de un conjunto de espacios de señales, operadores lineales y un conjunto de matrices especiales, donde los métodos algebraicos se utilizan para generar señales que se transforman como estimadores computacionales. Este punto es el más importante en todo el proyecto de investigación.

3. Utilizando las representaciones matriciales de cada transformada, se puede implementar cada transformada discreta en el lenguaje de programación MATLAB, utilizando un la herramienta computacional de cómputo en paralelo, llamada *Paralell Computing Toolbox*. Además, para un mejor manejo, se desarrolló un ambiente gráfico para el computo de dichas transformadas.

4. Como es de esperar a la hora de realizar cómputo en paralelo, los resultados numéricos permiten obtener una gran ventaja al realizar el cómputo en paralelo, en lugar de realizarlo de forma secuencial.

9. Recomendaciones

Los resultados de la investigación permiten plantear, muy respetuosamente, las siguientes recomendaciones:

- Tener disponibilidad de la computadora con más anticipación, para el desarrollo de las implementaciones en paralelo.
- Complementar la investigación en el caso de la TDF-1D para el caso de que la dimensión de la señal sea impar o potencia de 2, basado en el trabajo realizado por Pearse [36].

10. Apéndice

Los siguientes teoremas presenta algunas de las propiedades más importantes del producto de Kronecker. Estas propiedades se utilizan para demostrar los resultados teóricos. Dichos teoremas se pueden encontrar en [18, 36, 38]

Teorema 1 *Sea* $\mathbf{A} \in \mathbb{S}^{m\times n}$, $\mathbf{B} \in \mathbb{S}^{r\times s}$, $\mathbf{C} \in \mathbb{S}^{n\times p}$ *y* $\mathbf{D} \in \mathbb{C}^{s\times t}$. *Entonces*

- $(\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C})$.
- $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}$.
- $\mathbf{A} \otimes \mathbf{B} = (\mathbf{A} \otimes \mathbf{I}_r)(\mathbf{I}_n \otimes \mathbf{B})$.
- $\mathbf{I}_r \otimes \mathbf{I}_s = \mathbf{I}_{rs}$.

Teorema 2 *Sea* $n = rs$, $\mathbf{A} \in \mathbb{S}^{r\times r}$ *y* $\mathbf{B} \in \mathbb{S}^{s\times s}$. *Entonces* $\mathbf{A} \otimes \mathbf{B} = \mathbf{P}_r(\mathbf{B} \otimes \mathbf{A})\mathbf{P}_s$.

10.1. Demostración de resultados teóricos

Demostración de ecuación (3): Sea $\mathbf{x} \in \mathbb{S}^n$, $l_1, k_1 \in \mathbb{Z}_r$ y $l_2, k_2 \in \mathbb{Z}_s$. El vector $\mathbf{y} = (\mathbf{I}_s \otimes \mathbf{F}_r)\mathbf{P}_s\mathbf{x}$ es definido. Entonces

$$\mathbf{y}_{k_2 r+l_1} = \sum_{k_1\in\mathbb{Z}_r} \omega_r^{-k_1 l_1}\mathbf{x}_{sk_1+k_2}.$$

Ahora, considere $\mathbf{z} = \mathbf{T}_r\mathbf{y}$. Ahora, tenemos que $\omega_r^{-k_1 l_1} = \omega_n^{-sk_1 l_1}$, entonces

$$\begin{aligned}
\mathbf{z}_{k_2 r+l_1} &= \omega_n^{-k_2 l_1}\mathbf{y}_{k_2 r+l_1} \\
&= \omega_n^{-k_2 l_1}\sum_{k_1\in\mathbb{Z}_r} \omega_n^{-sk_1 l_1}\mathbf{x}_{sk_1+k_2} \\
&= \sum_{k_1\in\mathbb{Z}_r} \omega_n^{-(sk_1 l_1+k_2 l_1)}\mathbf{x}_{sk_1+k_2}
\end{aligned}$$

Sea $\omega = (\mathbf{F}_s^d \otimes \mathbf{I}_r)\mathbf{z}$. Entonces

$$\begin{aligned}
\omega_{l_1+l_2 r} &= \sum_{k_2\in\mathbb{Z}_s} \omega_s^{-k_2 l_2}\mathbf{z}_{k_2 r+l_1} \\
&= \sum_{k_2\in\mathbb{Z}_s} \omega_s^{-k_2 l_2}\left(\sum_{k_1\in\mathbb{Z}_r} \omega_n^{-(sk_1 l_1+k_2 l_1)}\mathbf{x}_{sk_1+k_2}\right) \\
&= \sum_{k_2\in\mathbb{Z}_s} \omega_n^{-rk_2 l_2}\left(\sum_{k_1\in\mathbb{Z}_r} \omega_n^{-(sk_1 l_1+k_2 l_1)}\mathbf{x}_{sk_1+k_2}\right) \\
&= \sum_{k_2\in\mathbb{Z}_s}\left(\sum_{k_1\in\mathbb{Z}_r} \omega_n^{-(rk_2 l_2+sk_1 l_1+k_2 l_1)}\mathbf{x}_{sk_1+k_2}\right)
\end{aligned}$$

Pero, $sk_1l_1 + k_2l_1 \equiv (k_2 + k_1s)(l_1 + l_2r) \bmod n$, entonces

$$\begin{aligned} &= \sum_{k_2\in\mathbb{Z}_s}\left(\sum_{k_1\in\mathbb{Z}_r}\omega_n^{-(k_2+k_1s)(l_1+l_2r)}\mathbf{x}_{sk_1+k_2}\right) \\ &= \sum_{k_2\in\mathbb{Z}_s}\sum_{k_1\in\mathbb{Z}_r}\omega_n^{-(k_2+k_1s)(l_1+l_2r)}\mathbf{x}_{sk_1+k_2}. \end{aligned}$$

Sea $m = sk_1 + k_2$, $k = l_1 + l_2r$ y $m, k \in \mathbb{Z}_n$ entonces $l_1, k_1 \in \mathbb{Z}_r$, $l_2k_2 \in \mathbb{Z}_s$ y $n = rs$. Entonces

$$\begin{aligned} &= \sum_{m\in\mathbb{Z}_n}\omega_n^{-mk}\mathbf{x}_m \\ &= \mathcal{F}(k) \end{aligned}$$

Demostración de ecuación (4): De [36], se sabe que $\mathbf{F}_{\mathbf{X}} = \mathcal{R}_{m,n}\left\{(\mathbf{F}_n \otimes \mathbf{F}_m)\mathcal{V}\{\mathbf{X}\}\right\}$. Entonces:

$$\begin{aligned} \mathbf{F}_m \otimes \mathbf{F}_n &= (\mathbf{F}_n \otimes \mathbf{I}_m)(\mathbf{I}_n \otimes \mathbf{F}_m) \\ &= \mathbf{P}_n(\mathbf{I}_m \otimes \mathbf{F}_n)\mathbf{P}_m(\mathbf{I}_n \otimes \mathbf{F}_m) \end{aligned}$$

Por lo tanto, se obtiene:

$$\mathbf{F}_{\mathbf{X}} = \mathcal{R}_{m,n}\left\{\mathbf{P}_n(\mathbf{I}_m \otimes \mathbf{F}_n)\mathbf{P}_m(\mathbf{I}_n \otimes \mathbf{F}_m)\mathcal{V}\{\mathbf{X}\}\right\}.$$

Demostración de ecuación (5): Sea $\mathbf{z} = \boldsymbol{L}_n^{n^2}\left(\mathbf{I}_n \otimes \mathbf{F}_n\right)\left(\mathbf{h} \odot (\mathbf{1}_n \otimes \mathbf{x})\right)$. Este vector se puede expresar de la siguiente manera

$$\mathbf{z} = \mathbf{L}_n^{n^2} \bigsqcup_{m\in\mathbb{Z}_n} \mathbf{s}_m, \tag{7}$$

donde $\mathbf{s}_m \in \mathbb{C}^n$, such that $\mathbf{s}_m = \mathbf{F}_n\left([\mathbf{H}(m,:)]^T \odot \mathbf{x}\right)$. Aplicando el operador $\mathcal{R}_{n,n}$ a (7), we obtain

$$\begin{aligned} \mathcal{R}_{n,n}\{\mathbf{z}\} &= \mathcal{R}_{n,n}\left\{\mathbf{L}_n^{n^2} \bigsqcup_{m\in\mathbb{Z}_n} \mathbf{s}_m\right\} \\ &= \left(\mathcal{R}_{n,n}\left\{\bigsqcup_{m\in\mathbb{Z}_N} \mathbf{s}_m\right\}\right)^T. \end{aligned}$$

Sea $\mathbf{S} \in \mathbb{C}^{n\times n}$ such that $\mathbf{S} = \mathcal{R}_{n,n}\left\{\bigsqcup_{m\in\mathbb{Z}_N} \mathbf{s}_m\right\}$. Then

$$\begin{aligned} \mathcal{R}_{n,n}\{\mathbf{z}\}(m,k) &= \mathbf{S}^T(m,k) \\ &= \mathbf{s}_m(k) \\ &= \sum_{j\in\mathbb{Z}_N}\mathbf{x}(j)\mathbf{H}(m,k)\omega_n^{-jk}. \end{aligned}$$

Finalmente, si realizamos el producto Hadamard de $\mathbf{A}$ y $\mathbf{S}^T$, entonces obtenemos

$$\begin{aligned} \tfrac{1}{\sqrt{n}}(\mathbf{A} \odot \mathbf{S}^T)(m,k) &= \tfrac{1}{\sqrt{n}}\, \mathbf{A}(m,k) \cdot \mathbf{S}^T(m,k) \\ &= \tfrac{1}{\sqrt{n}}\, \mathbf{A}(m,k) \cdot \mathbf{s}_m(k) \\ &= \mathbf{T}_{\mathbf{x}}(m,k) \end{aligned}$$

Demostración de ecuación (6): Utilizando la representación matricial de la $2D$-DFT, la matriz DCD se puede expresar como

$$\mathbf{C}_{x,y} = \mathcal{R}_{n,n}\{(\mathbf{F}_n \otimes \mathbf{F}_n)\mathcal{V}\{(\mathbf{A}_{x,y} \odot \phi)^T\}\}, \tag{8}$$

Consideremos lo siguiente:

1. Usando la propiedad $\mathcal{V}\{\mathbf{R}^T\} = L_n^{n^2}\mathcal{V}\{\mathbf{R}\}$, tenemos:

   $$\mathcal{V}\{(\mathbf{A}_{x,y} \odot \phi)^T\} = \mathbf{L}_n^{n^2}\mathcal{V}\{(\mathbf{A}_{x,y} \odot \phi)\}.$$

   Por tanto se obtiene que

   $$\mathbf{C}_{x,y} = \mathcal{R}_{n,n}\{(\mathbf{F}_n \otimes \mathbf{F}_n)\mathbf{L}_n^{n^2}\mathcal{V}\{(\mathbf{A}_{x,y} \odot \phi)\}\}. \tag{9}$$

2. Por el Teorema 1:

   $$\mathbf{F}_n \otimes \mathbf{F}_n = (\mathbf{I}_n \otimes \mathbf{F}_n)(\mathbf{F}_n \otimes \mathbf{I}_n).$$

   Por tanto, aplicando esta sustitución en la ecuación (9) se obtiene

   $$\mathbf{C}_{x,y} = \mathcal{R}_{n,n}\{(\mathbf{I}_n \otimes \mathbf{F}_n)(\mathbf{F}_n \otimes \mathbf{I}_n)\mathbf{L}_n^{n^2}\mathcal{V}\{(\mathbf{A}_{x,y} \odot \phi)\}\}. \tag{10}$$

3. Por el Teorema 2:

   $$\mathbf{F}_n \otimes \mathbf{I}_n = \mathbf{L}_n^{n^2}(\mathbf{I}_n \otimes \mathbf{F}_n)\mathbf{L}_n^{n^2}.$$

   Por tanto, aplicando esta sustitución en la ecuación (10) se obtiene

   $$\mathbf{C}_{x,y} = \mathcal{R}_{n,n}\{(\mathbf{I}_n \otimes \mathbf{F}_n)\mathbf{L}_n^{n^2}(\mathbf{I}_n \otimes \mathbf{F}_n)\mathbf{L}_n^{n^2}\mathbf{L}_n^{n^2}\mathcal{V}\{(\mathbf{A}_{x,y} \odot \phi)\}\}. \tag{11}$$

4. Usando la propiedad $\mathbf{L}_n^{n^2}\mathbf{L}_n^{n^2} = \mathbf{I}_{n^2}$ y aplicando esta igualdad en la ecuación (11) se obtiene

   $$\mathbf{C}_{x,y} = \mathcal{R}_{n,n}\{(\mathbf{I}_n \otimes \mathbf{F}_n)\mathbf{L}_n^{n^2}(\mathbf{I}_n \otimes \mathbf{F}_n)\mathcal{V}\{(\mathbf{A}_{x,y} \odot \phi)\}\}.$$

Referencias

[1] G. S. Almasi and A. Gottlieb. *Highly Parallel Computing*. Benjamin-Cummings Publishing Co, 1989.

[2] L. Auslander and R. Tolimieri. Computing decimated finite cross-ambiguity functions. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 36(3):359–364, 1988.

[3] Louis Auslander and R. Tolimier. *On finite Gabor expansion of signals*, pages 13–23. Springer-Verlag, London, UK, 1990.

[4] J.J. Benedetto and J.J. Donatelli. Ambiguity function and frame-theoretic properties of periodic zero-autocorrelation waveforms. *IEEE Journal of Selected Topics in Signal Processing*, 1(1):6–20, june 2007.

[5] J.J. Benedetto and J.J. Donatelli. Frames and a vector-valued ambiguity function. In *42nd Asilomar Conference on Signals, Systems and Computers*, pages 8–12, oct. 2008.

[6] B. Boashash. *Time Frequency Analysis*. Elsevier Science, 2003.

[7] Helmut Bolcskei and Franz Hlawatsch. Discrete Zak transforms, polyphase transforms, and applications. *IEEE Trans. Signal Processing*, 45:851–866, 1997.

[8] L. Brandolini, L. Colzani, A. Iosevich, and G. Travaglini. *Fourier Analysis and Convexity*. Applied and Numerical Harmonic Analysis. Birkhäuser Boston, 2011.

[9] S.A. Broughton and K.M. Bryan. *Discrete Fourier Analysis and Wavelets: Applications to Signal and Image Processing*. Wiley, 2011.

[10] C. Candan, M.A. Kutay, and H.M. Ozaktas. The discrete fractional fourier transform. *Signal Processing, IEEE Transactions on*, 48(5):1329–1337, May 2000.

[11] Michael Clausen and Ulrich Baum. Fast fourier transforms for symmetric groups: Theory and implementation. *Mathematics of Computation*, 61(204), 1993.

[12] Leon Cohen. *Time-frequency analysis: theory and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[13] M. Drubin. Kronecker product factorization of the fft matrix. *Computers, IEEE Transactions on*, C-20(5):590 – 593, may 1971.

[14] T.A. Ell and S.J. Sangwine. Hypercomplex fourier transforms of color images. *Image Processing, IEEE Transactions on*, 16(1):22–35, Jan 2007.

[15] S. Engelberg. *Digital Signal Processing: An Experimental Approach*. Signals and Communication Technology. Springer, 2008.

[16] H.G. Feichtinger and T. Strohmer. *Advances in Gabor Analysis*. Applied and Numerical Harmonic Analysis. Birkhäuser Boston, 2002.

[17] Franz Franchetti, Markus Puschel, Yevgen Voronenko, Srinivas Chellappa, and Jose M. F. Moura. Discrete Fourier transform on multicore. *IEEE Signal Processing Magazine, special issue on "Signal Processing on Platforms with Multiple Cores"*, 26(6):90–102, 2009.

[18] A. Graham. *Kronecker products and matrix calculus: with applications*. Ellis Horwood series in mathematics and its applications. Horwood, 1981.

[19] K. Gröchenig. *Foundations of Time-Frequency Analysis*. Applied and Numerical Harmonic Analysis. Birkhäuser Boston, 2001.

[20] F. Hlawatsch and G.F. Boudreaux-Bartels. Linear and quadratic time-frequency signal representations. *Signal Processing Magazine, IEEE*, 9(2):21 –67, April 1992.

[21] MATLAB: GUI. `http://au.mathworks.com/discovery/matlab-gui.html`, 3 de junio del 2015.

[22] MATLAB: Parallel Computing Toolbox. `http://au.mathworks.com/help/distcomp/`, 3 de junio del 2015.

[23] M. Petrou and C. Petrou. *Image Processing: The Fundamentals*. John Wiley & Sons, 2010.

[24] M. R. Portnoff. Time-frequency representation of digital signals and systems based on shorttime fourier analysis. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(1), 1980.

[25] P. A. Regalia and S. K. Mitra. Kronecker products, unitary matrices, and signal processing applications. *SIAM Rev.*, 31:586–613, December 1989.

[26] M.S. Richman, T.W. Parks, and R.G. Shenoy. Discrete-time, discrete-frequency, time-frequency analysis. *Signal Processing, IEEE Transactions on*, 46(6):1517 –1527, June 1998.

[27] D. Rodriguez. A computational kronecker-core array algebra sar raw data generation modeling system. In *Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on*, volume 1, pages 116 –120 vol.1, 2001.

[28] D. Rodriguez, J. Seguel, and E. Cruz. Algebraic methods for the analysis and design of time-frequency signal processing algorithms. In *Circuits and Systems, 1993., ISCAS '93, 1993 IEEE International Symposium on*, pages 196 –199 vol.1, may 1993.

[29] Stephen J. Sangwine and Todd A. Ell. Complex and hypercomplex discrete fourier transforms based on matrix exponential form of euler formula. *Applied Mathematics and Computation*, 219(2):644 – 655, 2012.

[30] S.W. Smith. *Digital Signal Processing: A Practical Guide for Engineers and Scientists*. Demystifying technology series : by engineers, for engineers. Newnes, 2003.

[31] Juan Pablo Soto Quiros and Domingo Rodriguez. Representación matricial de algoritmos en paralelo de la transformada discreta de fourier, la función discreta de ambiguedad y la distribución discreta de cohen. *La Gaceta (La Real Sociedad de la Matemática Española)*, 17(3), 2013.

[32] P. Soto-Quiros. A mathematical framework for parallel computing of discrete-time discrete-frequency transforms in multi-core processors. *Applied Mathematics and Information Sciences*, 8(6):2795–2801, 2014. cited By 0.

[33] P. Soto-Quiros. Application of block matrix theory to obtain the inverse transform of the vector-valued dft. *Applied Mathematical Sciences*, 9(52):2567–2577, 2015. cited By 0.

[34] P. Soto-Quiros. A parallel framework with block matrices of a discrete fourier transform for vector-valued discrete-time signals. *En revisión*, 2015.

[35] M. O. Tokhi, M. A. Hossain, and M. H. Shaheed. *Parallel computing for real-time signal processing and control*. Springer, 2003.

[36] Richard Tolimieri, Myoung An, and Chao Lu. *Algorithms for Discrete Fourier Transform and Convolution (Signal Processing and Digital Filtering)*. Springer, oct 1997.

[37] R. Trobec, M. VajterÅ¡ic, and P. Zinterhof. *Parallel computing: numerics, applications, and trends*. Springer, 2009.

[38] C. Van Loan. *Computational Framework of the Fast Fourier Transform*. SIAM, 1992.

[39] Charles F. van Loan. The ubiquitous kronecker product. *J. Comput. Appl. Math.*, 123:85–100, November 2000.

[40] D.F. Walnut. *An Introduction to Wavelet Analysis*. Applied and Numerical Harmonic Analysis. Birkhäuser Boston, 2002.

[41] W. Wong. *Discrete Fourier Analysis*. Springer Basel, 2011.

[42] Xiang-Gen Xia. Discrete chirp-fourier transform and its application to chirp rate estimation. *Signal Processing, IEEE Transactions on*, 48(11):3122 –3133, nov 2000.

[43] Toshio Yoshizawa, Shigeki Hirobayashi, and Tadanobu Misawa. Noise reduction for periodic signals using high-resolution frequency analysis. *EURASIP Journal on Audio, Speech, and Music Processing*, 2011(1), 2011.