Instituto Tecnológico de Buenos Aires

Ingeniería Informática

Métodos Numéricos Avanzados

==========

Trabajo Practico Numero 2

Filtrado de imágenes utilizando transformada de Fourier discreta

Profesora Noni, Laura

Castiglione, Gonzalo

Wassington, Axel

Procesamiento de imágenes I++

Abstract

Este informe tiene como objetivo analizar espectralmente una imagen en escala de grises y utilizar la transformada discreta de Fourier para realizar filtrados espaciales de acuerdo a diversos filtros. Se presentarán diferentes tipos de filtros y el resultado de aplicarlo sobre una imagen.

1 Introducción

El procesamiento digital de imágenes es el conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad o facilitar la búsqueda de información.
El proceso de filtrado es el conjunto de técnicas englobadas dentro del preprocesamiento de imágenes cuyo objetivo fundamental es obtener, a partir de una imagen origen, otra final cuyo resultado sea más adecuado para una aplicación específica mejorando ciertas características de la misma que posibilite efectuar operaciones del procesado sobre ella.
Los principales objetivos que se persiguen con la aplicación de filtros son:

- Suavizar la imagen: reducir la cantidad de variaciones de intensidad entre píxeles vecinos.
- Eliminar ruido: eliminar aquellos píxeles cuyo nivel de intensidad es muy diferente al de sus vecinos y cuyo origen puede estar tanto en el proceso de adquisición de la imagen como en el de transmisión.
- Realzar bordes: destacar los bordes que se localizan en una imagen.
- Detectar bordes: detectar los píxeles donde se produce un cambio brusco en la función intensidad.

Por tanto, se consideran los filtros como operaciones que se aplican a los píxeles de una imagen digital para optimizarla, enfatizar cierta información o conseguir un efecto especial en ella.

El proceso de filtrado puede llevarse a cabo sobre los dominios de frecuencia y/o espacio.

2 Cuerpo del artículo

2.1 Implementación de un programa que computa la TDF 2D.

A continuación se presenta la ecuación de la Transformada Discreta de Fourier de una secuencia bidimensional $x_{n,m}$ de tamaño $N \times N$ (imagen), la cual se define como:

$$X_{l,k} = \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} x_{n,m} e^{-i\frac{2\pi}{N}(nl+mk)} \tag{1}$$

Siendo la transformada inversa discreta:

$$x_{n,m} = \frac{1}{N^2} \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} X_{l,k} e^{i\frac{2\pi}{N}(nl+mk)} \tag{2}$$

El cálculo de la transformada de Fourier de una matriz bidimensional, se realiza a partir de la fórmula (1). En donde, por cada posición (l, k), se calcula la sumatoria de los valores transformados de toda la matriz.

La implementación de este algoritmo puede encontrarse en la sección *Anexo*, el en punto (1).

Una implementación clásica, como la presentada, es inútil en cuanto a usos prácticos para una imagen de dimensiones considerables debido al tiempo que tarda en computarse. Lo más común es el uso de la transformada rápida de Fourier. Puede encontrarse la implementación de este algoritmo por ejemplo en *Matlab* u *Octave*[1] con el nombre de $fft2()$. Esta implementación utiliza una técnica de "*divide and conquer*" y esta además desarrollada en un lenguaje compilado, y de esa forma, se consigue una performance que no se puede conseguir utilizando las otras funciones que ofrecen estos dos lenguajes en forma nativa.

2.2 El archivo *saturno* contiene una matriz de 400×400 píxeles y corresponde a niveles de intensidad luminosa comprendidos entre 0 y 255.

Para visualizar esta imagen en escala de grises, es necesario establecer un mapa de color de 255 niveles.

Visualización la Figura que muestra una imagen del planeta Saturno, capturada por la misión Voyager.

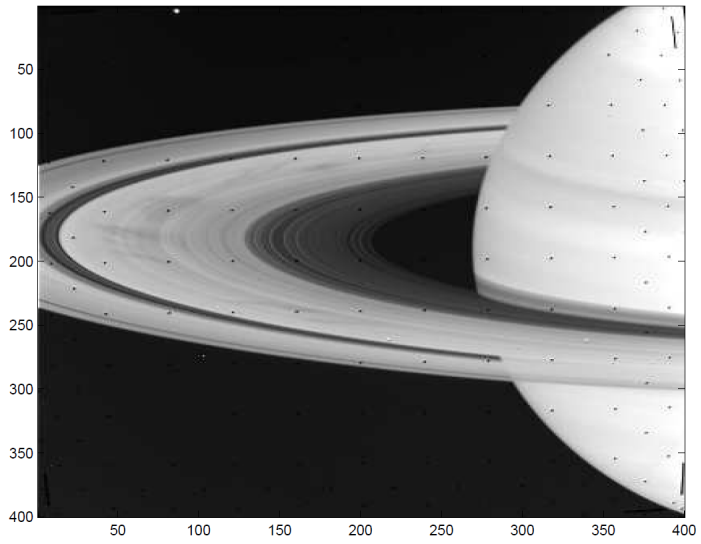

Figure 1: Imagen de saturno obtenida a partir de las mediciones

2.3 Computar la Transformada discreta de Fourier de la imagen original. Armar las imágenes de 400×400 píxeles correspondientes a la amplitud y la fase.

Un problema a considerar previamente a computar las imágenes correspondientes, es que la imagen que se esta buscando debe ser construida en escala de grises $[0, 255]$ y las matrices obtenidas por la transformada de fourier

[1] En el presente informe se utilizará la versión 3.6.4 de GNU Octave

(tanto en el ángulo de fase así como en la amplitud) pueden tomar valores fuera de ese intervalo.

Para el primer caso, para el cálculo de la amplitud, dada X (transformada de fourier de x), se utilizó el módulo del número complejo obtenido sobre cada valor de X.

Para normalizarlo se utilizaron dos técnicas.

La primer técnica consiste en realizar un mapeo lineal entre el intervalo de valores que toma la matriz transformada de Fourier y el intervalo $[0 - 255]$ utilizando la siguiente fórmula:

$$Output_i = floor((\frac{absol_i}{maxAbsol}) * 255) \tag{3}$$

Donde $maxAbsol$ es el máximo de las amplitudes, $absol$ es cada una de las amplitudes y $Output$ es el píxel de salida.

Como se puede ver en la fórmula (3), para lograr dicho mapeo, primero se divide el intervalo original por el máximo valor absoluto (se lo lleva a $[0, 1]$), luego se multiplica por 255 para dejarlo en el intervalo deseado. El problema con esto es que todavía se tienen valores reales (aunque dentro del intervalo), mientras que la escala exige valores enteros. Es por esto último que a todo lo obtenido anteriormente, se le calcula el $floor$.

Una vez realizado el procedimiento, la imagen obtenida es la siguiente:

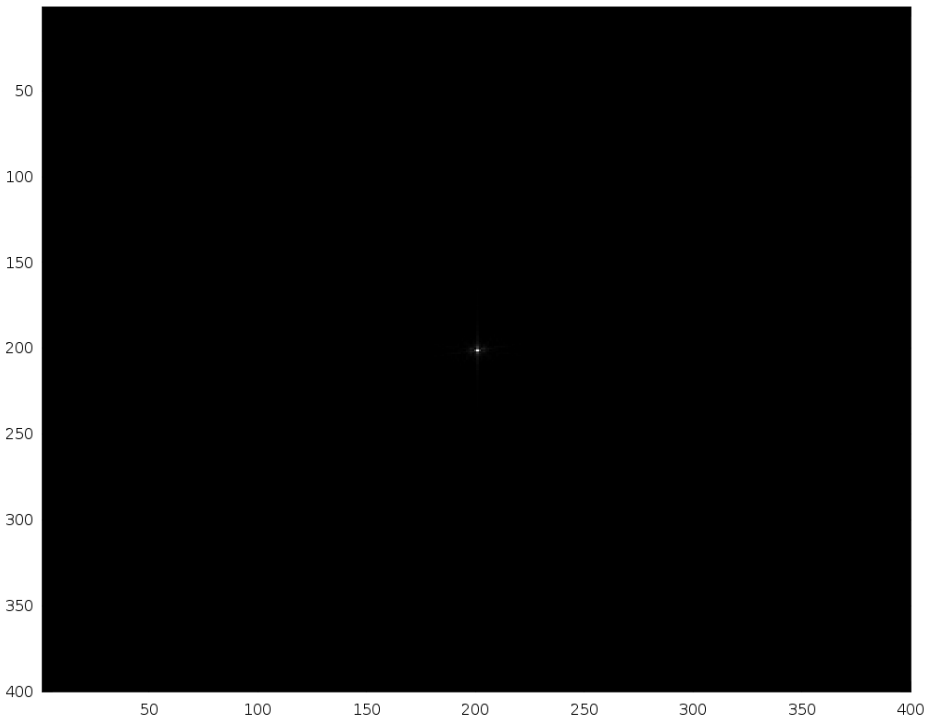

Figure 2: Imagen de la amplitud de cada píxel de la matriz transformada de fourier de x

Como se puede observar en la figura 2, hay un grupo de frecuencias (centro de la figura) muy superiores que el resto. Pero debido a la técnica de mapeo utilizada, no se puede decir mucho más al respecto.

En cambio, utilizando una escala logarítmica:

$$Output_i = \frac{255}{log(1 + Absol_i)} \tag{4}$$

Donde *absol* es cada una de las amplitudes y *Output* es el píxel de salida.
Con este nuevo procedimiento, la imagen que se obtiene es la siguiente:

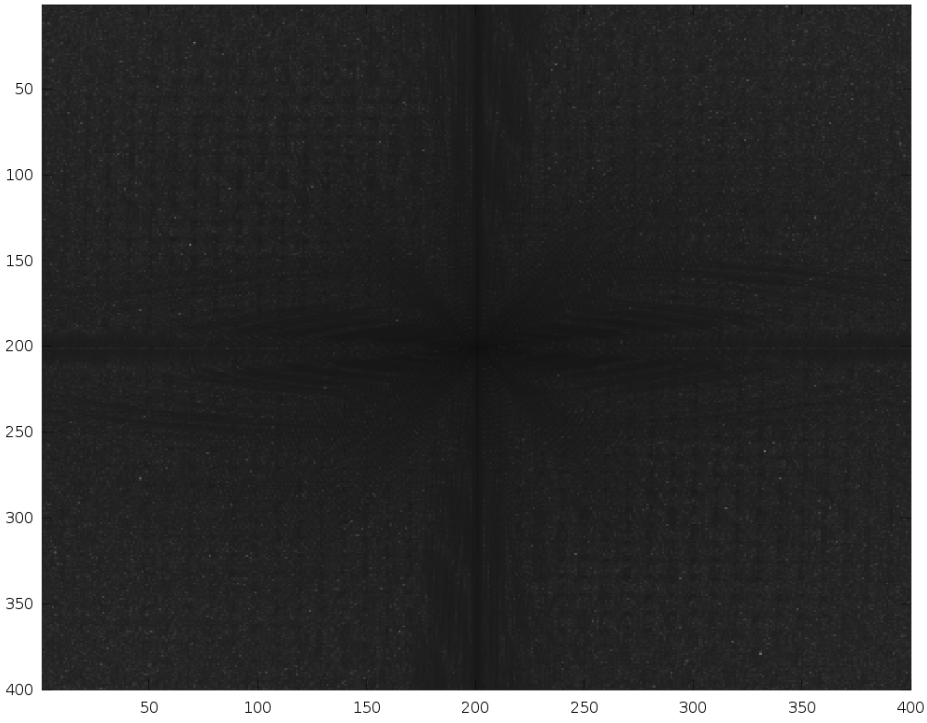

Figure 3: Imagen logarítmica de la amplitud de cada píxel de la matriz transformada de fourier de x

Aplicando dicha fórmula, se obtiene una visualización más interesante ya que se puede distinguir (además del pico de frecuencia central) una trama en el resto de las frecuencias.

Nota: Ambas imágenes de la amplitud de la frecuencia están "shifteadas" para que la frecuencia 0 *quede en el centro. Es decir,* se parte la imagen en dos columnas y se cambia el orden. Luego se parte la imagen en dos filas y nuevamente se cambia el orden. El código para este cambio se presenta en la sección *Anexo*, en el punto 2.

Para el segundo caso, para el cálculo de la fase, se utilizó el argumento del número complejo obtenido sobre cada valor de X.

Para normalizarlo se utilizó la primera técnica de mapeo, la de mapeo lineal:

$$Output_i = floor(((\frac{args_i}{2 * maxArg}) + 1/2) * 255) \tag{5}$$

Donde $maxArg$ es el argumento de mayor valor absoluto, $args$ es cada uno de los argumentos y $Output$ es el píxel de salida.

Como se puede ver en la fórmula (5), para lograr dicho mapeo, primero se divide el intervalo original por dos veces el máximo valor absoluto (se lo lleva a $[-0.5, 0.5]$), luego se suma 0.5 y multiplica por 255 para dejarlo en el intervalo deseado. El problema con esto es que todavía se tienen valores reales (aunque dentro del intervalo), mientras que la escala exige valores enteros. Es por esto último que a todo lo obtenido anteriormente, se le calcula el $floor$.

La imagen obtenida fue la siguiente:

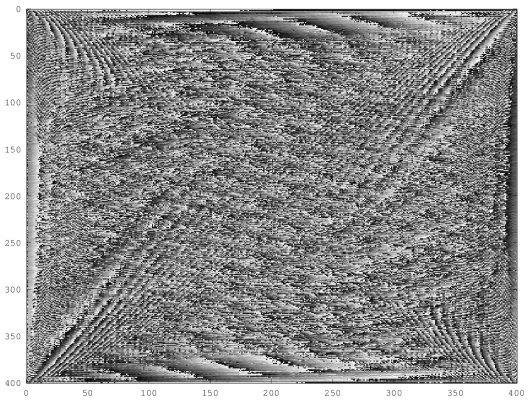

Figure 4: Imagen de la fase de cada píxel de la matriz transformada de fourier de x

La matriz con los valores obtenidos para la amplitud y para la fase, pueden encontrarse en los archivos *absolutos.mat* y *argumentos.mat* respectivamente.

2.4 Considerar el efecto que producen los siguientes filtros $H_{k,l}$ de 400×400 píxeles en el dominio de las frecuencias (espaciales):

Para la aplicación de los filtros en la frecuencia: Se transformó la señal a la frecuencia. Se multiplicó la señal transformada por el filtro punto a punto. Se destransformó la señal filtrada. Y se tomó la parte real de la señal filtrada. El proceso se describe de acuerdo a la siguiente imagen[2]:

[2]Imagen extraída de un artículo de wikipedia: `http://commons.wikimedia.org/wiki/File:Etapas_filtrado_dominio_frecuencia.jpg`

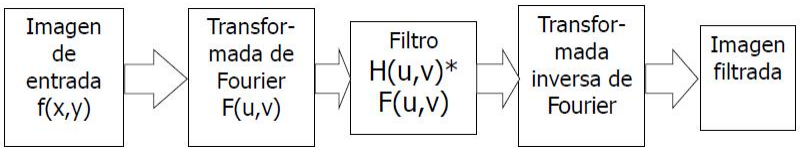

Figure 5: Proceso de filtrado de una imagen utilizando filtros en dominio de frecuencia

El código utilizado para ejecutar este proceso se presenta en la sección *Anexo*, en el punto 3.

Debido a la simplicidad de las condiciones de cada filtro, el código utilizado no se presenta sobre este informe pero puede ser encontrado en los archivos $filtro1.m$, $filtroGauss.m$ y $filtroDamero.m$

2.4.1 Filtro 1

$$H_{k,l} = \begin{cases} 0 & 0 \leq k \leq 400,\ 190 \leq l \leq 210 \\ 1 & 0 \leq l \leq 400, 190 \leq k \leq 210 \\ 1 & \text{, en todo otro caso} \end{cases}$$

Resultado:

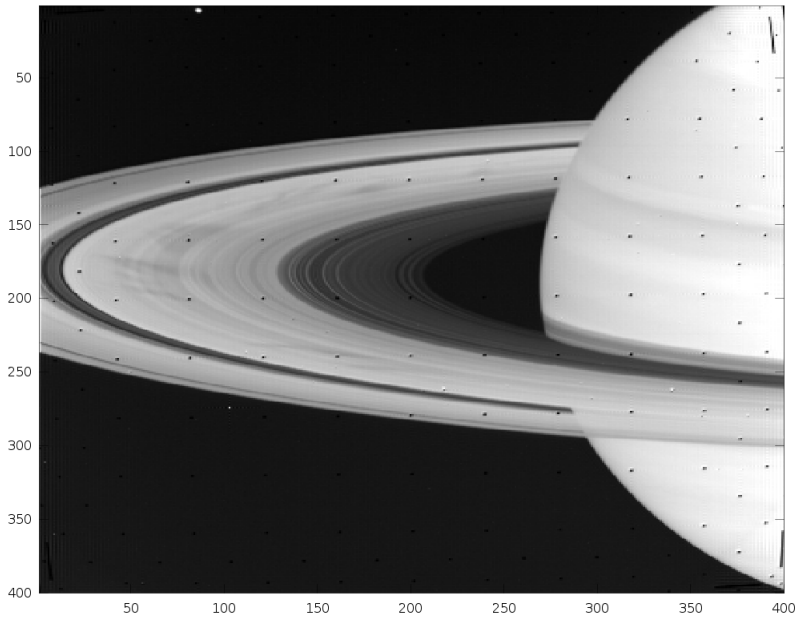

Figure 6: Imagen obtenida luego aplicado el filtro 1 sobre la matriz transformada

También conocido como "filtro notch". Este filtro no permite el paso de señales cuyas frecuencias se encuentran comprendidas entre las frecuencias de corte superior e inferior. Se utiliza para eliminar interferencia destructiva,

encontrar patrones en una imagen o eliminar patrones. En este caso se puede ver (sobre todo en los puntos negros y blancos) como se repite el patrón cruz a lo largo y ancho de la imagen.

2.4.2 Filtro Gaussiano

$$H_{k,l} = exp(-0.1(k^2 + l^2))$$

Resultado:

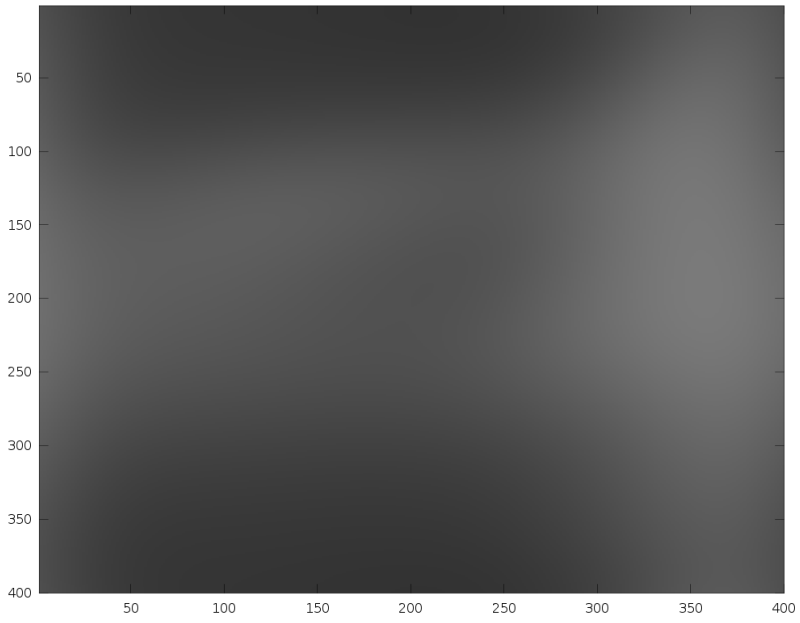

Figure 7: Imagen obtenida luego aplicado el filtro Gaussiano sobre la matriz transformada

También conocido como "desenfoque gausiano". Este filtro mezcla ligeramente los colores de los píxeles que estén vecinos el uno al otro, en un mapa de bits (imagen), haciendo que la imagen pierda algunos detalles minúsculos, y de esta forma se vea más suave. Se uso un desenfoque muy exagerado, haciendo que apenas se distinga la zona más clara de la imagen de la más oscura. Justamente, una de las utilidades de este filtro es la búsqueda de bordes.

2.4.3 Filtro Dámero

$$H_{k,l} = \begin{cases} 0 & l+k \text{ es par} \\ 1 & l+k \text{ es impar} \end{cases}$$

Resultado:

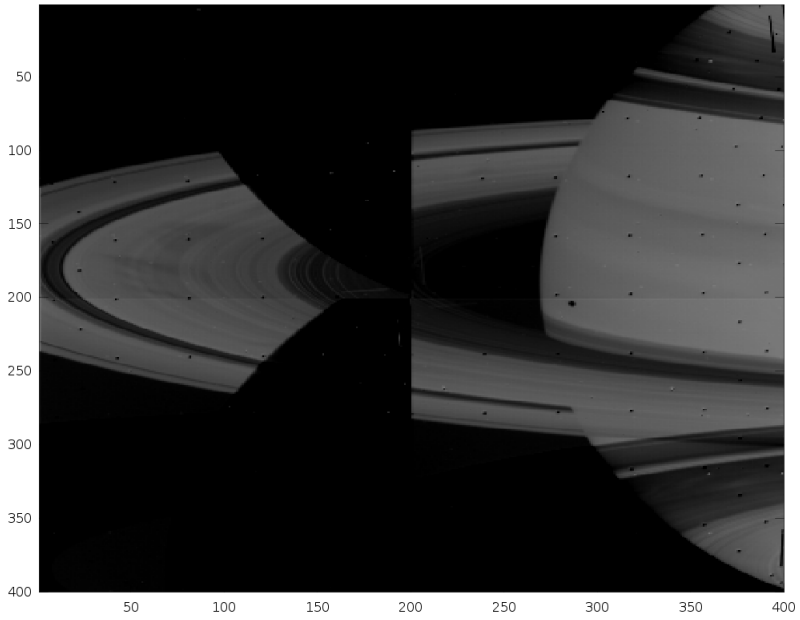

Figure 8: Imagen obtenida luego aplicado el filtro Dámero sobre la matriz transformada

3 Conclusiones

Luego de analizados los resultados obtenidos con la transformada discreta de fourier. Resulta muy notable la cantidad de áreas en donde esta técnica podría tener aplicaciones ya que permite realizar filtrado de imágenes. Especialmente en el área de la informática.

Un problema que se encontró en esta implementación del algoritmo, es la notable cantidad de procesamiento requerido para la aplicación de filtros para una imagen relativamente pequeña (encima en blanco y negro!). Es por eso que se recomienda el uso de la FFT (trasformada rápida de Fourier).

Los filtros pueden tener variadas utilidades. Se pueden usar para hacer desde efectos sencillos a efectos complejos. Muchos son conocidos y se conocen muchas de sus utilidades, pero otros son nuevos y aún tienen utilidades desconocidas o incomprensibles.

El filtrado de imágenes resulta un tema complejo e interesante.

4 Bibliografía

- Digital Filters - `http://homepages.inf.ed.ac.uk/rbf/HIPR2/filtops.htm`
- Digital Filter - `http://en.wikipedia.org/wiki/Digital_filter`
- Band-stop Filter - `http://en.wikipedia.org/wiki/Band-stop_filter`
- Gaussian Blur - `http://en.wikipedia.org/wiki/Gaussian_blur`

5 Anexo

5.1 Cálculo de transformada de Fourier discreta

Lo que esta pieza de código realiza es iterar por cada uno de los píxeles (l, k) de la matriz de transformación de fourier, y calcula su valor de acuerdo a la sumatoria por cada píxel n, m con los valores transformados de la matriz original. Esta implementación asume que la matriz es cuadrada de NxN

```
function transform = tdf2d(matrix)
    N = rows(matrix);
    transform = zeros(N,N);
    aux = -i * 2 * pi / N;
    for l = 1 : N
        for k = 1 : N
            for n = 1 : N
                for m = 1 : N
                    c = e ** (aux * ((n-1) * (l-1) + (m-1) * (k-1)));
                    transform(l, k) += matrix(n, m) * c;
                endfor
            endfor
        endfor
    endfor
endfunction
```

5.2 Shifteo de la *frecuencia* de una matriz, a fines de centrar la frecuencia 0 en el centro

```
//shifteado de las columnas.
Y = [Y(:,201:400) Y(:,1:200)]
//shifteado de las filas.
Y = [Y(201:400,:) ; Y(1:200,:)]
```

5.3 Código para aplicación de un filtro en dominio de frecuencia

```
//Se transformó la señal a la frecuencia.
Y = tdf2d(X);
//Se multiplicó la señal transformada por el filtro punto a punto.
Y2 = Y .* filtroxxx(400);
//Se destransformó la señal filtrada.
X2 = tdf2dInv(Y2);
//Se tomó la parte real de la señal filtrada.
Output = real(X2);
```