Tecnológico de Costa Rica
Escuela Ingeniería Electrónica
Procesamiento Digital de Señales

Prof. Jose Miguel Barboza

Students:

Kendall González León 2015087861

Carlos Peralta Coto 201237200

Tarea 2

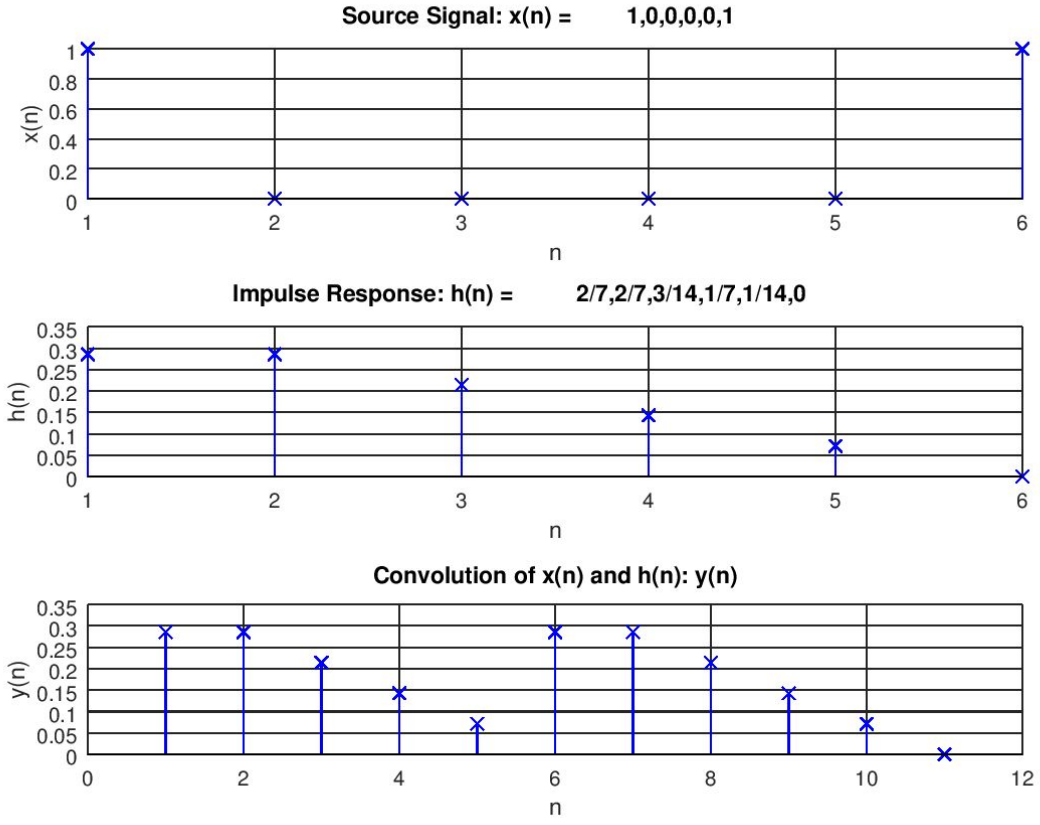
Semestre 1, 2018

Índice

Ejercicio 1

Para resolver este ejercicio, primero se creo una figura, con su respectivo nombre, posteriormente se definió una función ‘grapherDiscrete’, la cual realiza gráficos de forma genérica, la misma genera un gráfico en el dominio discreto con su título, color y figura personalizado, con cuadricula y ejes rotulados. Esto ya que los gráficos se usarán bastante en este ejercicio y en el resto de la tarea en general.

Primero, se definió la señal de prueba, la cual es $x(n) = \{1, 0, 0, 0, 0, 1\}$. Luego, se define la respuesta al impulso $h(n) = \{2/7, 2/7, 3/14, 1/7, 1/14, 0\}$. Luego se procede a graficar cada una de las señales requeridas: $x(n)$, $h(n)$ y $y(n) = x(n) * h(n)$.



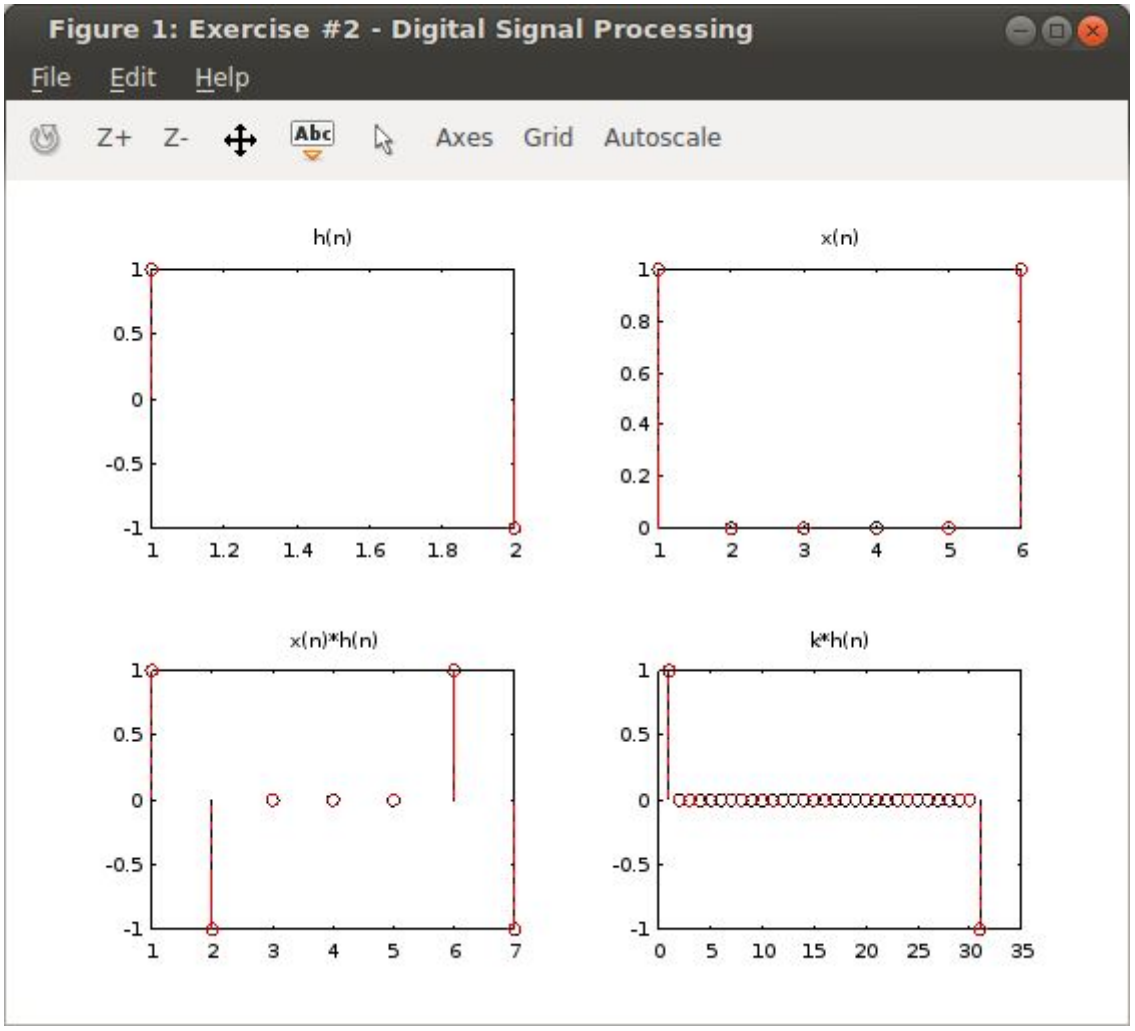
La respuesta al impulso dada es LPF (Low Pass Filter) FIR (Finite Impulse Response), esta conclusión se obtuvo, aplicando la Transformada Rápida de Fourier, en dicha representación se obtiene que las componentes de frecuencia más bajas poseen magnitudes mayores y se disminuye conforme la frecuencia aumenta.

Ejercicio 2

Para resolver este ejercicio se creó una figura que contuviera las siguientes gráficas:

- *h(n)*,
- *x(n)*,
- *x(n)*h(n)*,
- *k*h(n)*,

en donde *h(n)* es la respuesta al impulso, *x(n)* es la entrada al sistema, *x(n)*h(n)* es la convolución de ambas señales y *k* es una señal constante. Para esta última gráfica, no se toman en cuenta ni la primer ni la última señal (1 y 31) debido a que no representan la salida de la convolución con una señal constante.



Como se muestra en k*h(n), la salida de la convolución de una señal constante con la respuesta al impulso h(n) es una señal constante de 0. Este tipo de filtro es paso alto, debido a que al restar dos muestras sucesivas atenuamos la señal en los puntos en los que varía lentamente (bajas frecuencias) y acentuamos allí donde la señal varía rápidamente (altas frecuencias).

Ejercicio 3

Se procedió de igual forma al ejercicio número 1. Para este ejercicio se definió la función ‘my_conv’ la misma recibe como parámetro dos vectores. Es necesario determinar las caracteristicas de la convolución. Dada una susesión $x(n)$ y una respuesta al impulso $h(n)$.

- El número mínimo de muestra de la convolución está dado por: $y_{min} = x_{min} + h_{min}$ y para el caso del ejercicio, es 0.
- El número máximo de muestra de la convolución está dado por: $y_{max} = x_{max} + h_{max}$.
- La longitud de la señal resultante de la convolución será $N = y_{max} - 1$.
- La convolución es conmutativa.

La convolución está dada por

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = x(n) * h(n)$$

de la definición anterior se obtiene lo siguiente:

- La complejidad del algoritmo es $O(n^2)$.
- se deduce que habrán dos iteradores anidados, uno para situarse en la posición de cada muestra de la señal resultante y otro para calcular el valor en dicha posición, en cual requiere realizar multiplicaciones.

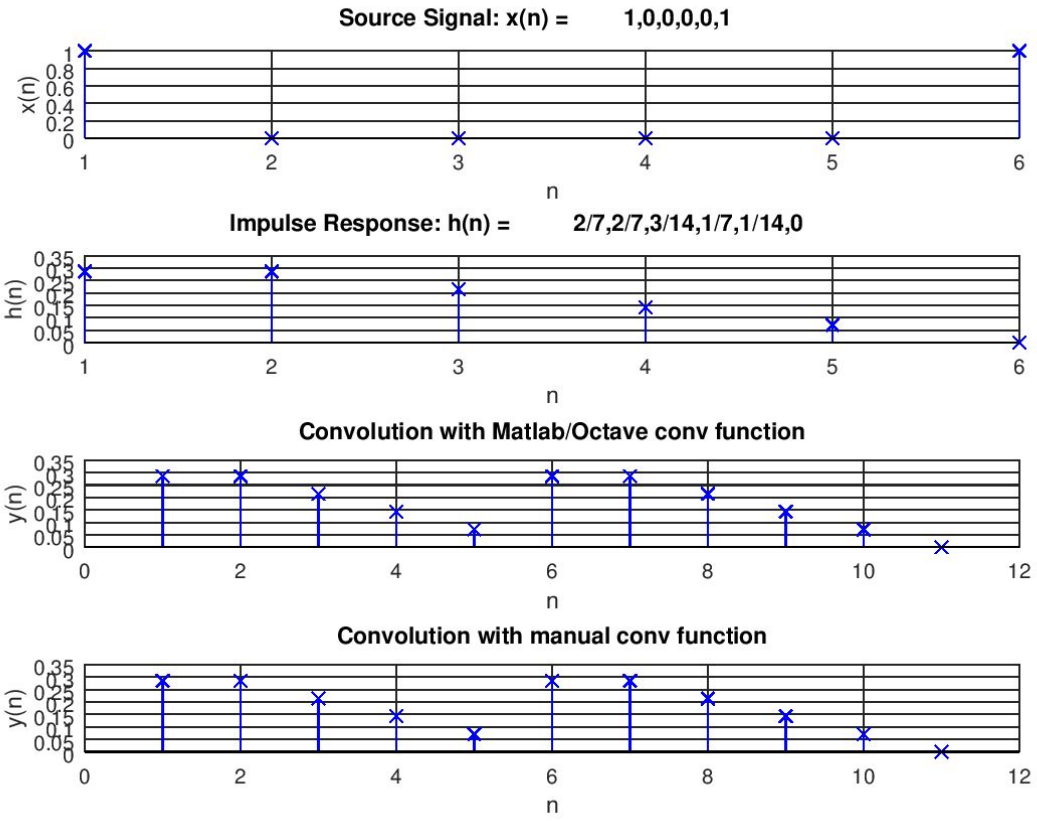
Descripción del algoritmo implementado

- Se define un vector que almacenará la salida del algoritmo, este será modificado en cada iteración.
- Se define un vector por cada entrada del algoritmo, de tal forma que ambos vectores tengan la misma cardinalidad, esto debido a que las operaciones implementadas en Octave/Matlab sobre vectores, como la suma, resta, multiplicación, etc, requiere que la longitud de ambos operandos sea la misma. Esto se debe realizar ya que no hay garantía de que las señales $x(n)$ y $h(n)$ tengan la misma longitud.
- Se define un iterador ‘for’, el cual iterará desde 1 hasta N. Es importante recalcar que los elementos de los vectores en Octave/Matlab están indexados comenzando desde 1 y no desde 0, como los lenguajes de programación tradicionales. Este ciclo, se encargará de definir cada valor de la salida.
- Se define otro iterador ‘for’, el cual recorrerá el vector que representa la señal de entrada, de misma forma, desde 1 hasta x_{max}.
- Debido a que se requiere obtener valores que no podrían estar definidos en el vector (posiciones no definidas, como por ejemplo, menores que cero) se define una validación tal que para operar el índice de la secuencia desde ser mayor a 0. En el caso que sea menor o igual no se toma ninguna acción, ya que se entiende que existen ceros en dichas posiciones y la multiplicación por 0 no afectaría el resultado.

- Se establecerá el valor de la posición actual en el vector de salida, donde se realiza la inversión, desplazamiento y multiplicación de los vectores.

Ejemplo

Se puede apreciar en la siguiente figura, la equivalencia de los algoritmos conv de Matlab/Octave y nuestro algoritmo.



Ejercicio 4

Para este ejercicio, se definió la función "*result=y(a_k, b_k, x, N, condi)*", en donde los parámetros son:

- **a_k**: vector con los coeficientes a_k
- **b_k**: vector con los coeficientes b_k
- **x**: referencia al archivo en donde está la función de entrada
- **N**: cantidad de salidas que se desean obtener
- **condi**: parámetro opcional, es un vector con las condiciones iniciales

Descripción del algoritmo implementado

- Primero se verifica si se ingresaron las condiciones iniciales, o no. Esto se realiza con *nargin*, que contiene el número de argumentos ingresados a la función. En caso de que no se ingresaran las condiciones iniciales, se crea un vector con ceros.
- Después se genera un vector que va a contener las respuestas, y se ingresa a un ciclo que va a iterar *N* veces. En cada uno de estos ciclos, se genera una respuesta *y(n)*.
- Se obtiene la respuesta de $b_k x(n-k)$ para todos los b_k ingresados en *b_k*.
- Se obtiene la respuesta de $a_k y(n-k)$ para todos los ak ingresados en *a_k*. En caso de que *n-k* sea un valor negativo, se utilizan las condiciones iniciales. De otra forma se utilizan los valores almacenados en el vector de resultados.
- Se grafica el resultado y se imprime.

Ejemplo

Se utilizan los siguiente valores de prueba:

- a_k = [1, -3, -4];
- b_k = [1, -2];
- x =

```
function f=x_funcion(n)
  if(n == 0)
    f = 1;
  else
    f = 0;
  endif
endfunction
```

- N = 5
- condi = [1, 2, 3, 4];

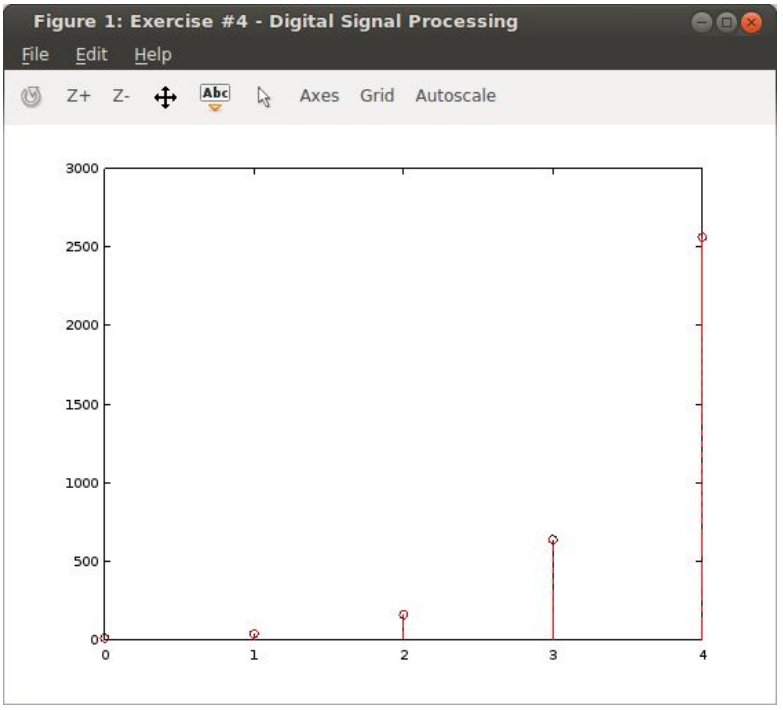
Para generar la salida de esta función, se utiliza el comando

```
>> y(a_k,b_k,@x_funcion ,5, condi);
```

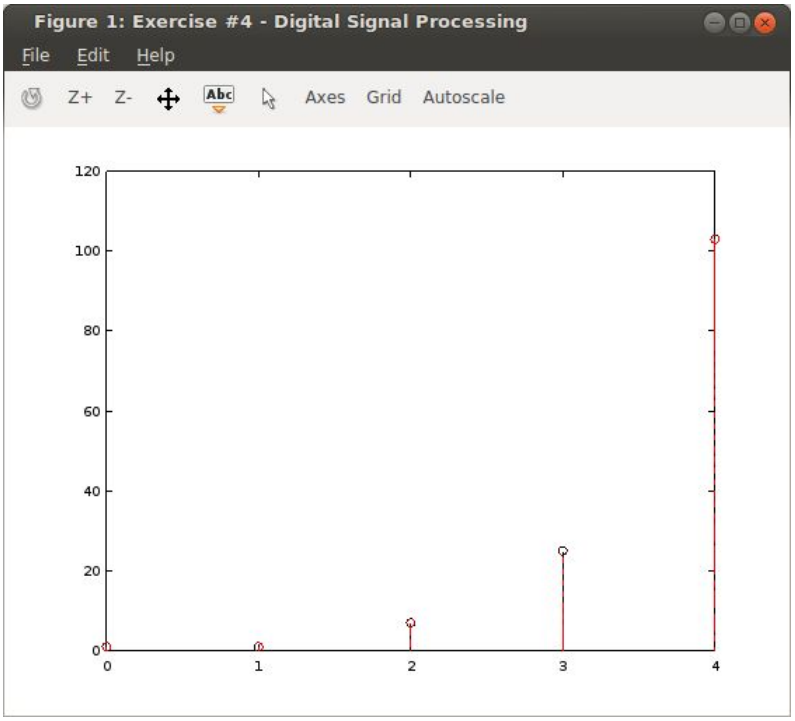
en donde @x_funcion es la referencia al archivo en donde se encuentra definida x, y se envían las condiciones iniciales. En caso de que no se deseen las condiciones iniciales, el comando a utilizar es

```
>> y(a_k,b_k,@x_funcion ,5);
```

El resultado con condiciones iniciales es:

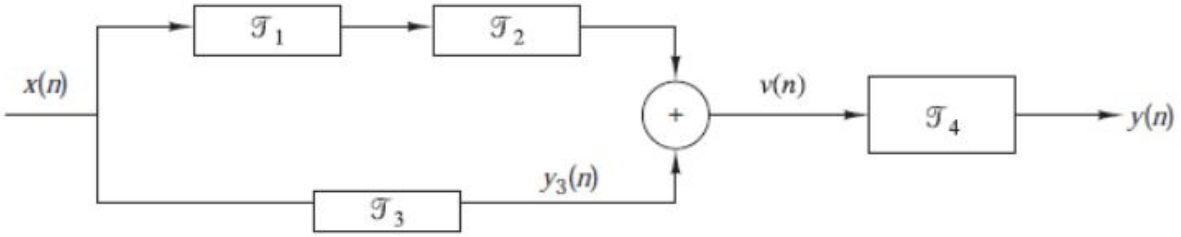


y sin condiciones iniciales es:



Ejercicio 5.

Se debe calcular la respuesta al impulso del sistema dado por el siguiente diagrama de bloques.

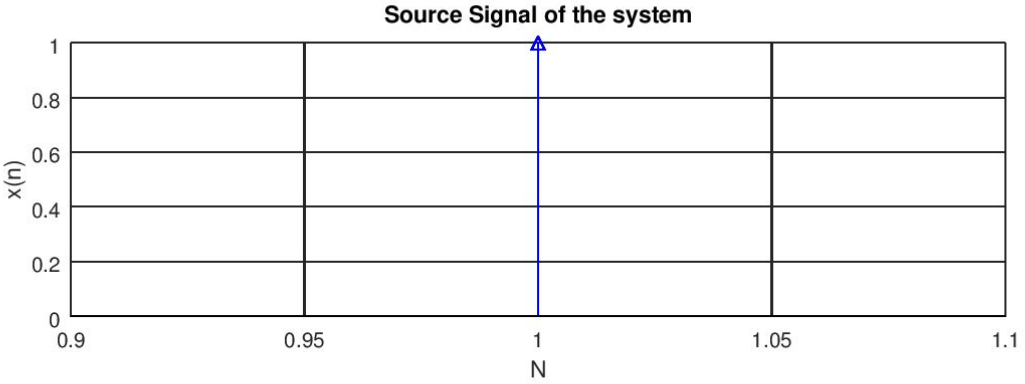


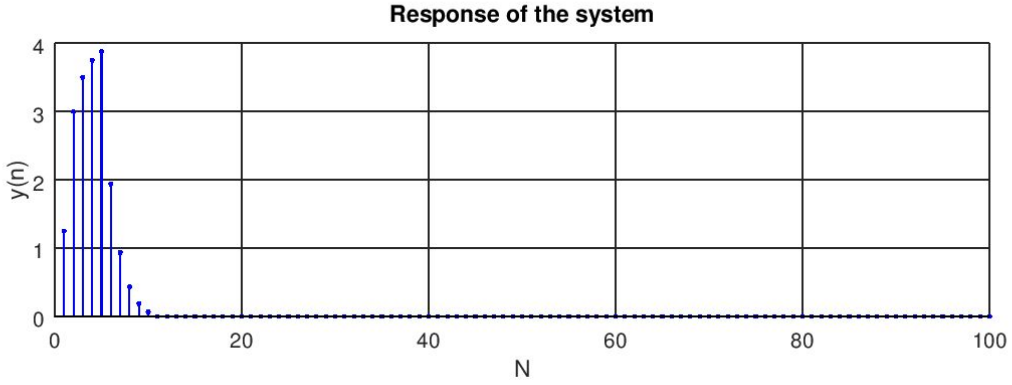
Se determinó que el sistema funciona de la siguiente forma:

- Sea A_1 la salida del subsistema T_1, está dado por la convolución de la señal de entrada $x(n)$ y su respuesta al impulso $h_1(n)$.
- Sea A_2 la salida del subsistema T_2, está dado por la convolución de A_1 y su respuesta al impulso $h_2(n)$.
- Sea A_3 la salida del subsistema T_3, está dado por la combinación lineal de la entrada actual, la entrada pasada y la entrada antepasada en cualquier posición.
- Sea V la señal $v(n)$, la cual está dada por la suma de A_2 y A_3.
- Sea A_4 la señal de salida $y(n)$, la cual está dada por la combinación lineal de la salida pasada, salida antepasada (existe retroalimentación) y la entrada actual proveniente de V y la señal pasada $V(n-1)$.

Cada subsistema tiene su respectiva función. Es necesario indicar que los subsistemas que dependen de entradas o salidas anteriores, deben tener memoria para poder operar las salidas actuales. Todos estos subsistemas están gobernados por la función System.

Para encontrar la respuesta al impulso del Sistema, según la teoría, será la salida del sistema ante una entrada del tipo Delta Dirac. El la siguiente figura se observa la respuesta al impulso del sistema.

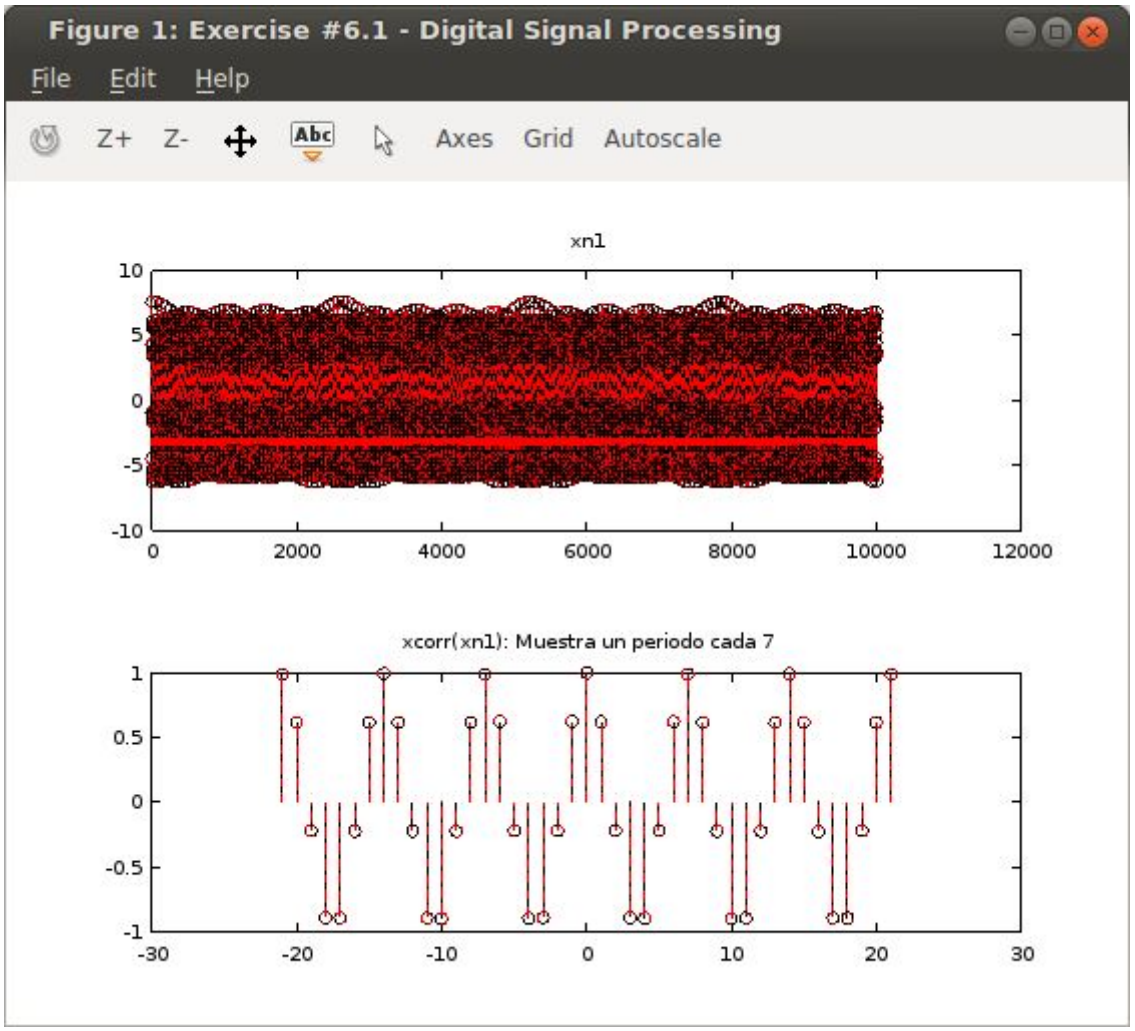

Source Signal of the system
x(n)
N
0
0.2
0.4
0.6
0.8
1
0.9
0.95
1
1.05
1.1


Response of the system
y(n)
N
0
1
2
3
4
0
20
40
60
80
100

Ejercicio 6

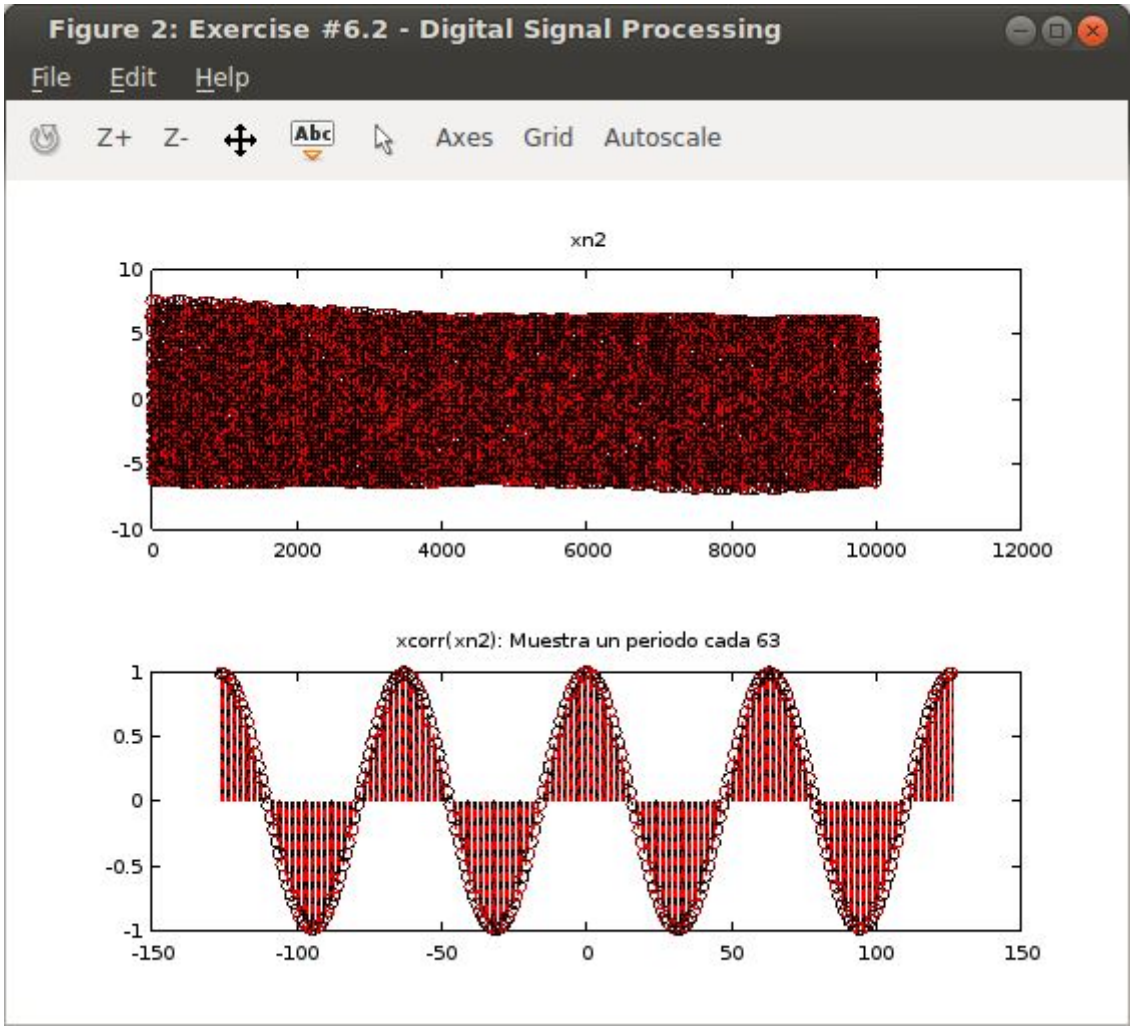
Para realizar este ejercicio, se utilizó la función xcorr en cada una de las imágenes, utilizando distintos límites de maxlag, para poder notar cada cuanto se encontraba el período señal. A continuación se muestra cada figura y el resultado de la autocorrelación.

Señal 1



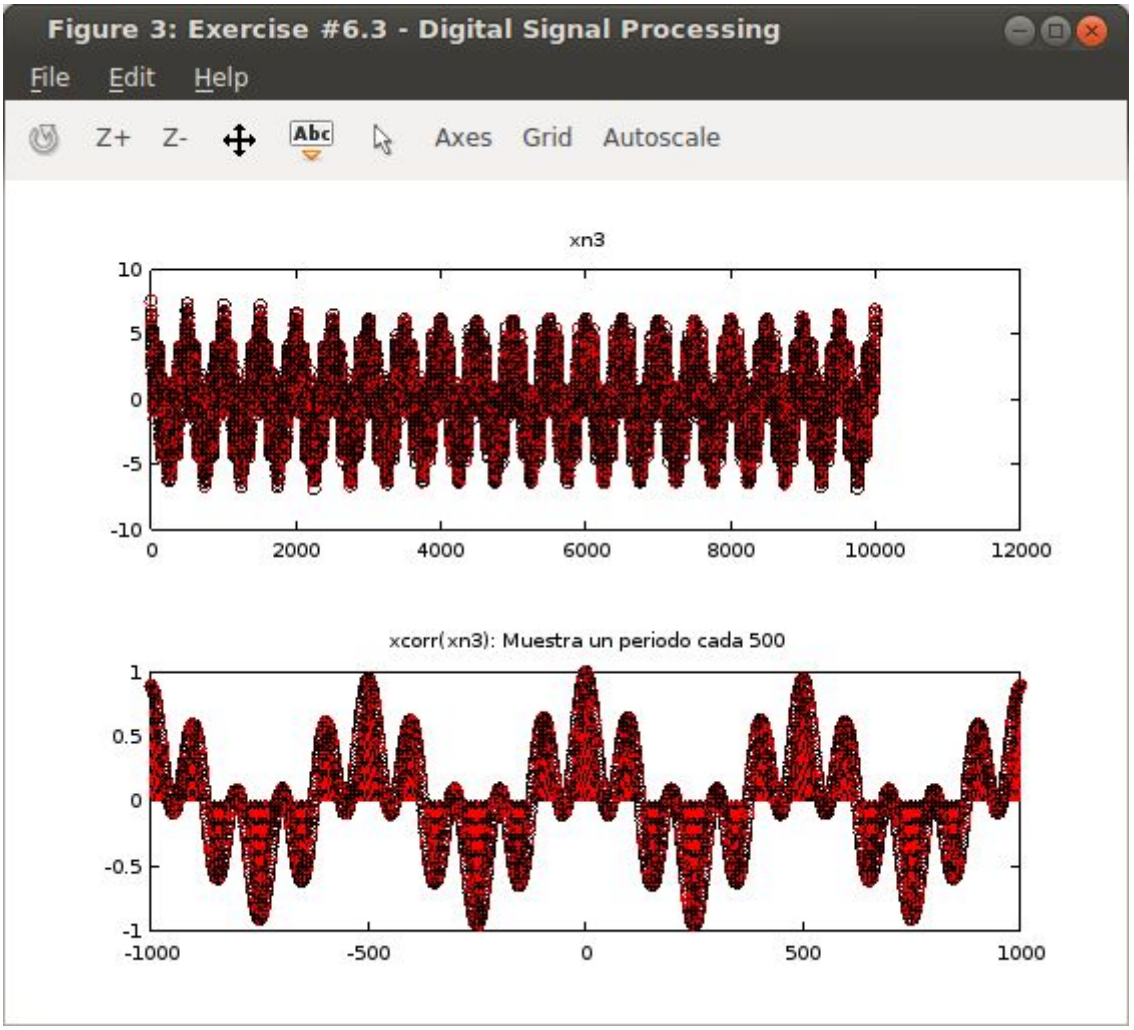
Se muestra claramente que el período de la señal está cada 7.

Señal 2



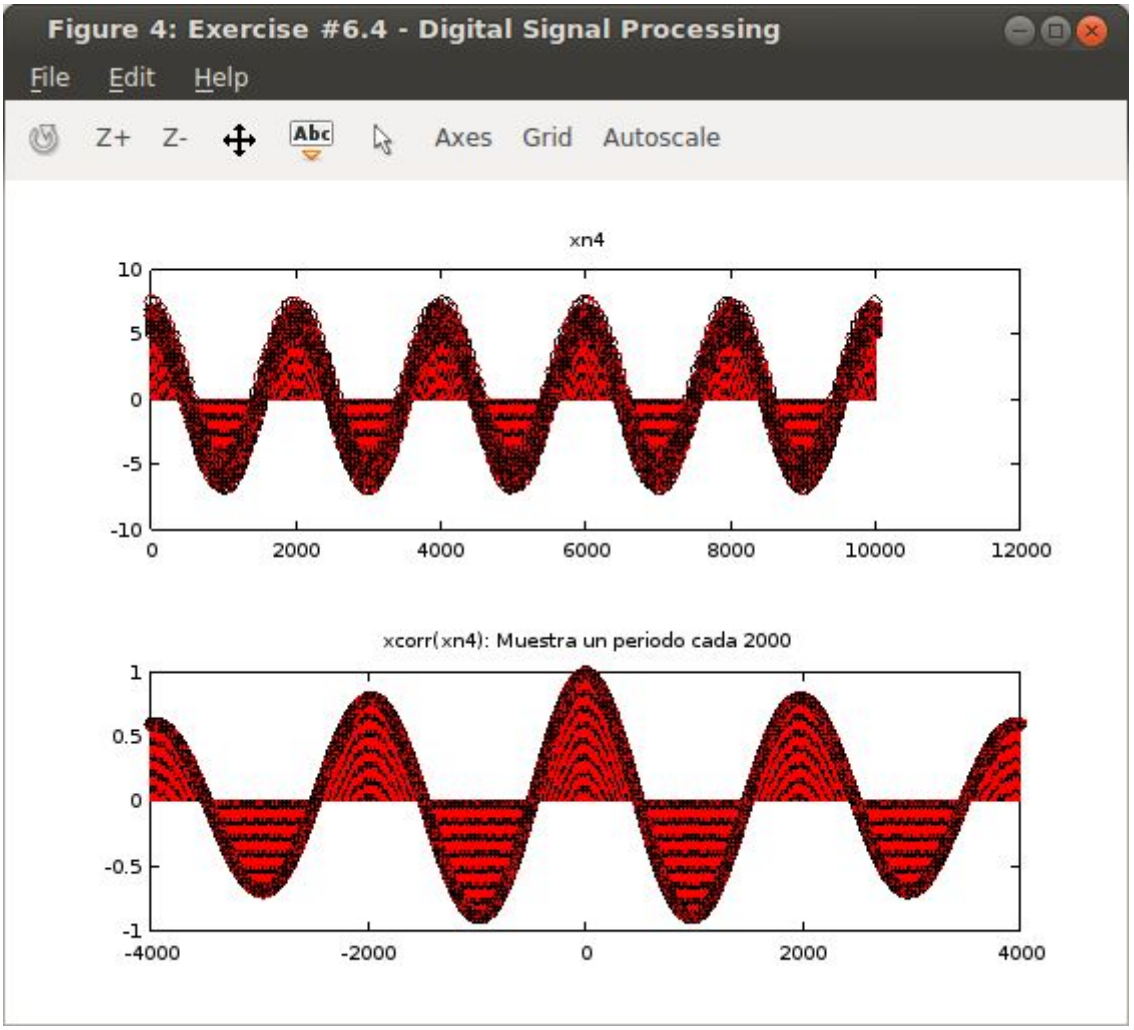
Se muestra claramente que el período de la señal está cada 63.

Señal 3



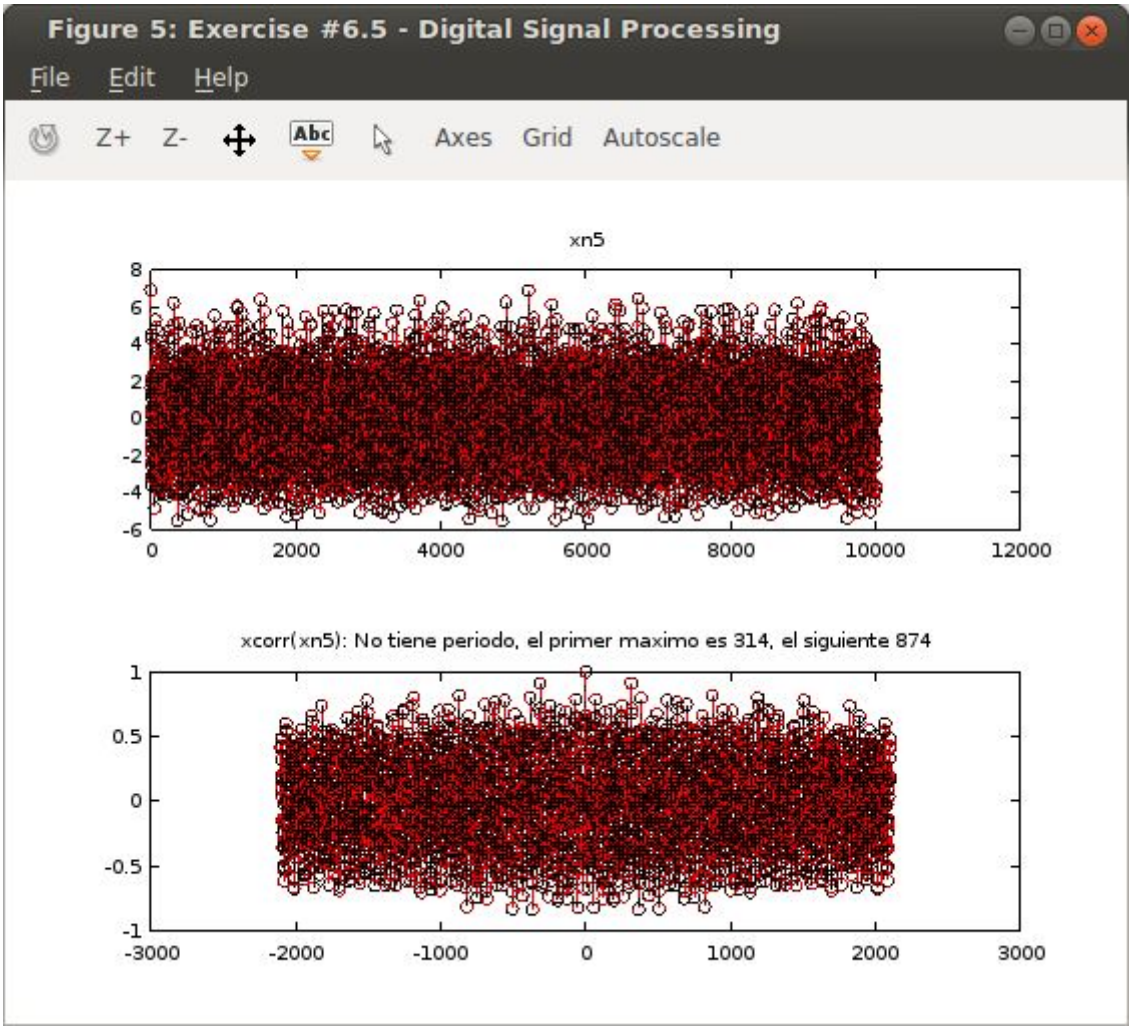
Se muestra claramente que el período de la señal está cada 500.

Señal 4



Se muestra claramente que el período de la señal está cada 2000.

Señal 5



Se muestra que a pesar de que tiene cierto rango en que parece que se están repitiendo los valores máximos, no hay ningún período de una señal presente en el ruido, o no tiene un período real.