Reporte 02A.- Simulación de Circuito RC

Alumno: Monroy Martos Elioth
Profesor: Gutierrez Aldana Eduardo
Materia: Teoría de Comunicaciones y Señales
Grupo: 3CM6

20 de noviembre de 2017

Índice

1. Introducción

Los filtros pasabajas, son un tipo de filtro los cuales permiten que las frecuencias menores a una frecuencia de corte (fc) "pasen" normalmente y las posteriores a esta son filtradas (es decir, no aparecen en la salida). Teóricamente, estos filtros deben de rechazar las frecuencias mayores a la frecuencia de corte, pero en la práctica, estos filtros atenúan las frecuencias mayores a la frecuencia de corte a cierta velocidad de decaimiento según sea el orden del filtro (1er orden-20dB/década, 2do orden-40dB/década, etc). Los filtros ideales no son posibles de implementar, pero existen diversas implementaciones de filtros que buscan acercase lo más posible a un filtro ideal.
La implementación más sencilla de un filtro pasabajas es la obtenida con un circuito RC (Resistencia-Capacitor), el cual es representado en la Figura 1.

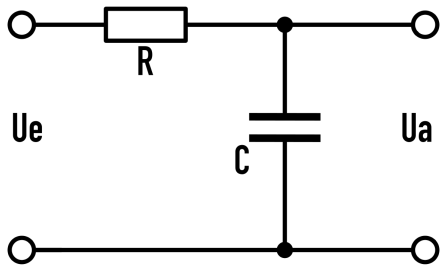

Figura 1: Circuito RC

La forma del filtro pasabajas en frecuencia, se puede observar en la Figura 2.

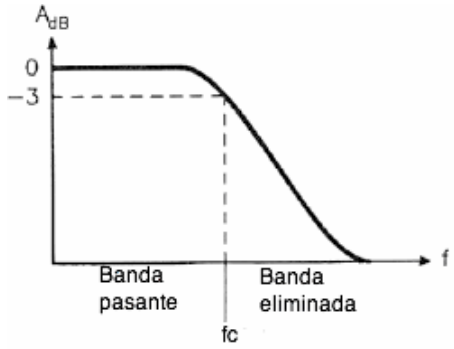

Figura 2: Diagrama de Bode para un filtro pasabajas

Donde la frecuencia de corte esta dada por:

$$w_c = \frac{1}{RC}$$
$$w_c = 2\pi f_c$$

Como se puede observar, la frecuencia de corte depende enteramente de la resistencia y capacitor, a esta clase de filtros también se le conoce como pasivos, ya que están compuestos puramente por elementos de este tipo.
Al analizar el circuito, se obtiene que la función de transferencia del mismo es:

$$H_{(s)} = \frac{1}{1+RsC}$$

Donde s, es un número complejo; s=jw.
De la anterior función de transferencia, podemos obtener la respuesta al impulso del circuito (la inversa de la transformada de Laplace de la función de transferencia), esto representa la respuesta del circuito a una entrada de voltaje consistente en un impulso o función delta de Dirac.

$$\frac{1}{RC} e^{-\frac{1}{RC}t} u\left(t\right)$$

Si se sustituye $w_c = \frac{1}{RC}$ se obtiene lo siguiente:

$$w_c e^{-w_c t} u\left(t\right)$$

De tal manera, es posible modelar el filtro sin necesidad de proponer un valor para la resistencia y el capacitor, tan solo es necesario conocer la frecuencia de corte deseada. Esto resulta útil al momento de realizar un programa que simule este tipo de filtro. Esta simulación, se puede obtener mediante la convolución de dos señales (la señal de entrada y la respuesta al impulso del circuito RC).
La convolución es un operador matemático que transforma dos funciones f y g en una tercera función que representa la magnitud en la que se superponen f y una versión trasladada e invertida de g.

2. Código

El programa elaborado en esta práctica, consistió en simular un filtro pasabajas mediante la convolución de dos señales. Una de ellas recibida como un parámetro de entrada y la otra sería la respuesta al impulso de un circuito RC (filtro pasabajas), la cual fue definida dentro del programa (dentro de la función "calcularFiltro()"). Esto con el fin de simular un filtro pasabajas con una frecuencia de corte de 1000Hz.
A continuación, se presenta el código elaborado.
funciones.h:

```
#ifndef __FUNCIONES_H__
#define __FUNCIONES_H__
  //Librerías de C
  #include <stdio.h>
  #include <stdlib.h>
  #include <math.h>
  //Métodos
  void leerCabeceras(char**);
  void escribirArchivo(short*);
  //Cabeceras
  int chunkid;
  int chunksize;
  int format;
  int subchunk1id;
  int subchunk1size;
  short audioformat;
  short numchannels;
  int samplerate;
  int byterate;
  short blockalign;
  short bitspersample;
  int subchunk2id;
  int subchunk2size;
  //Archivo
  FILE* entrada;
  FILE* salida;
  //Variables para muestras
  short muestra;
  int total_muestras;
  short headers[37];
  //Filtro
  #define PI acos(-1.0)//Defino la constante PI
  #define TOTAL_COEFICIENTES 20
```

```
34   double filtro[TOTAL_COEFICIENTES];//Arreglo para el filtro
35   double suma_filtro;
36   double e;
37   void calcularFiltro();
38   void calcularNuevasMuestras(short*);
39 #endif
```

filtro-psb.c:

```
1  #include"funciones.h"
2  int main(int argc, char *argv[]){
3    //Leo las cabeceras
4    leerCabeceras(argv);
5    //Defino variables
6    int i=0;
7    total_muestras=subchunk2size/blockalign;
8    short muestras[total_muestras];
9    //Leo las muestras
10   while (feof(entrada) == 0)
11   {
12     if(i<total_muestras){
13       fread(&muestra,sizeof(short),1,entrada);
14       muestras[i]=muestra;
15       i++;
16     }else{
17       fread(&headers,sizeof(short),37,entrada);
18       break;
19     }
20   }
21   calcularFiltro();
22   calcularNuevasMuestras(muestras);
23 }
24 void leerCabeceras(char ** argv){
25   entrada = fopen(argv[1], "rb");
26   salida=fopen(argv[2],"wb");
27   if(!entrada) {
28     perror("\nFile opening failed");
29     exit(0);
30   }
31   fread(&chunkid,sizeof(int),1,entrada);
32   fread(&chunksize,sizeof(int),1,entrada);
33   fread(&format,sizeof(int),1,entrada);
34   fread(&subchunk1id,sizeof(int),1,entrada);
35   fread(&subchunk1size,sizeof(int),1,entrada);
36   fread(&audioformat,sizeof(short),1,entrada);
37   fread(&numchannels,sizeof(short),1,entrada);
```

```
38    fread(&samplerate,sizeof(int),1,entrada);
39    fread(&byterate,sizeof(int),1,entrada);
40    fread(&blockalign,sizeof(short),1,entrada);
41    fread(&bitspersample,sizeof(short),1,entrada);
42    fread(&subchunk2id,sizeof(int),1,entrada);
43    fread(&subchunk2size,sizeof(int),1,entrada);
44  }
45  void escribirArchivo(short* muestras){
46    //Escribo cabeceras del archivo
47    fwrite(&chunkid,sizeof(int),1,salida);
48    fwrite(&chunksize,sizeof(int),1,salida);
49    fwrite(&format,sizeof(int),1,salida);
50    fwrite(&subchunk1id,sizeof(int),1,salida);
51    fwrite(&subchunk1size,sizeof(int),1,salida);
52    fwrite(&audioformat,sizeof(short),1,salida);
53    fwrite(&numchannels,sizeof(short),1,salida);
54    fwrite(&samplerate,sizeof(int),1,salida);
55    fwrite(&byterate,sizeof(int),1,salida);
56    fwrite(&blockalign,sizeof(short),1,salida);
57    fwrite(&bitspersample,sizeof(short),1,salida);
58    fwrite(&subchunk2id,sizeof(int),1,salida);
59    fwrite(&subchunk2size,sizeof(int),1,salida);
60    //Ahora escribo las muestras
61    int i=0;
62    for(i=0;i<total_muestras;i++){
63      fwrite(&muestras[i],sizeof(short),1,salida);
64    }
65    //Y por último los headers de goldwave
66    for(i=0;i<37;i++){
67      fwrite(&headers[i],sizeof(short),1,salida);
68    }
69  }
70  void calcularFiltro(){
71    e=exp(1);//Aquí obtengo el valor de e
72    int i;
73    for (i= 0; i < 20; i++){
74      filtro[i]=(2000*PI)*pow(e,((-1)*2000*PI*i)/44100);
75      suma_filtro+=filtro[i];
76      printf("%f\n", filtro[i]);
77    }
78  }
79  void calcularNuevasMuestras(short* muestras){
80    //Algoritmo para la convolución usando Input Side Algorithm
81    short nuevas_muestras[total_muestras];
82    int i,j;
```

```
83    //Primero inicializar a cero el arreglo
84    for (i = 0; i < total_muestras; i++){
85      nuevas_muestras[i]=0;
86    }
87    //Ahora si el algoritmo
88    for (i = 0; i < total_muestras; i++){
89      for (j = 0; j < TOTAL_COEFICIENTES; j++){
90        nuevas_muestras[i+j]+=(muestras[i]*filtro[j])/suma_filtro;
91      }
92    }
93    //Escribo el archivo
94    escribirArchivo(nuevas_muestras);
95  }
```

La convolución entre las señales fue realizada mediante el Input Side Algorithm[1].

3. Pruebas

Para comprobar el funcionamiento del programa, se ingreso la siguiente función, en un archivo con una frecuencia de muestreo de 44100 muestras/s. cos(2*pi*t*(exp(log(20)+n/N*6.6))):

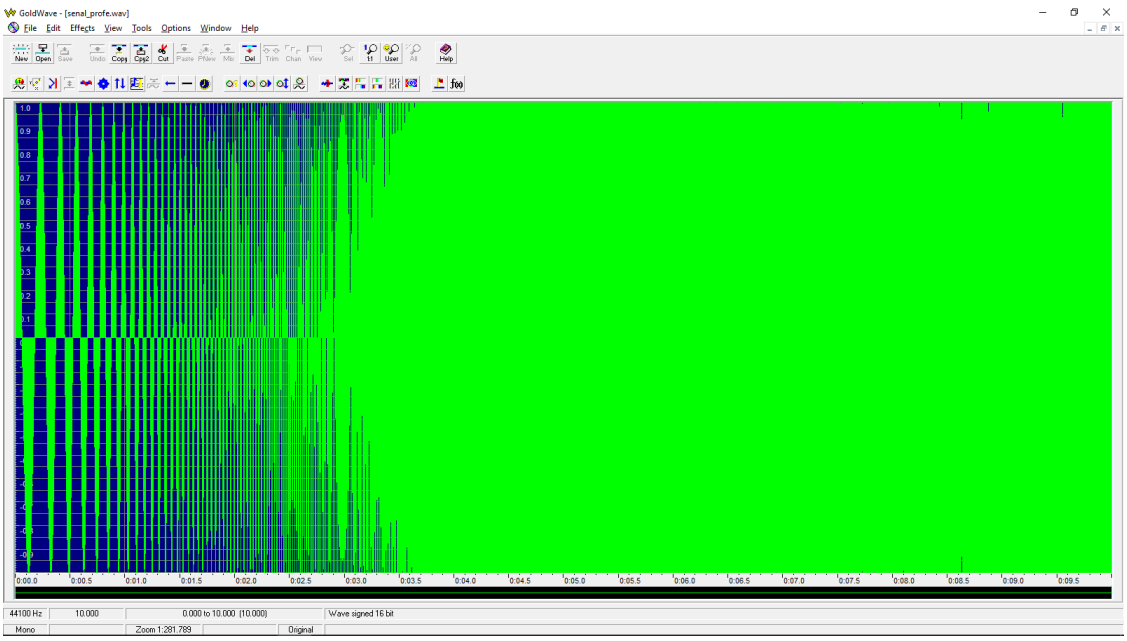

Figura 3: Entrada 1

Al realizar la convolución mediante el uso del programa, se obtuvo la siguiente salida:

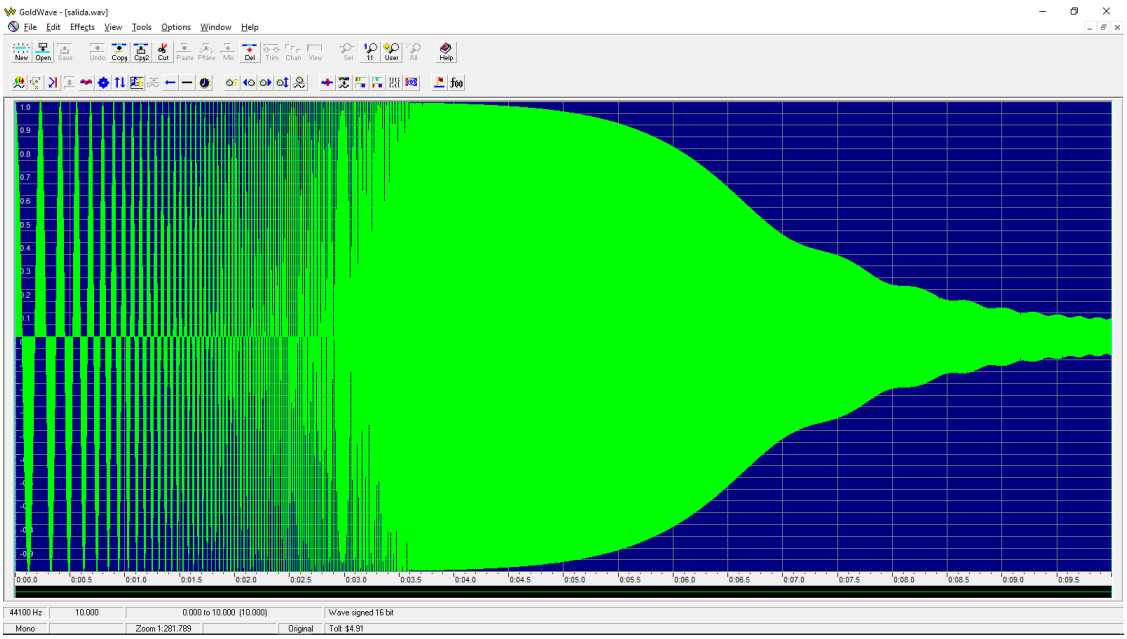

Figura 4: Salida 1 - 20 coeficientes en el filtro

Como se observa, la señal de salida es igual a la señal de entrada con la diferencia que cuando se alcanza la frecuencia de 1000Hz en la señal de salida, la amplitud de la misma empieza a decaer hasta aproximarse a cero. En la Figura 5, se observa la frecuencia que tiene la señal al momento de que esta decae 3dB o su amplitud es .707 veces la amplitud de la señal original. Esta frecuencia es la frecuencia de corte del filtro RC.

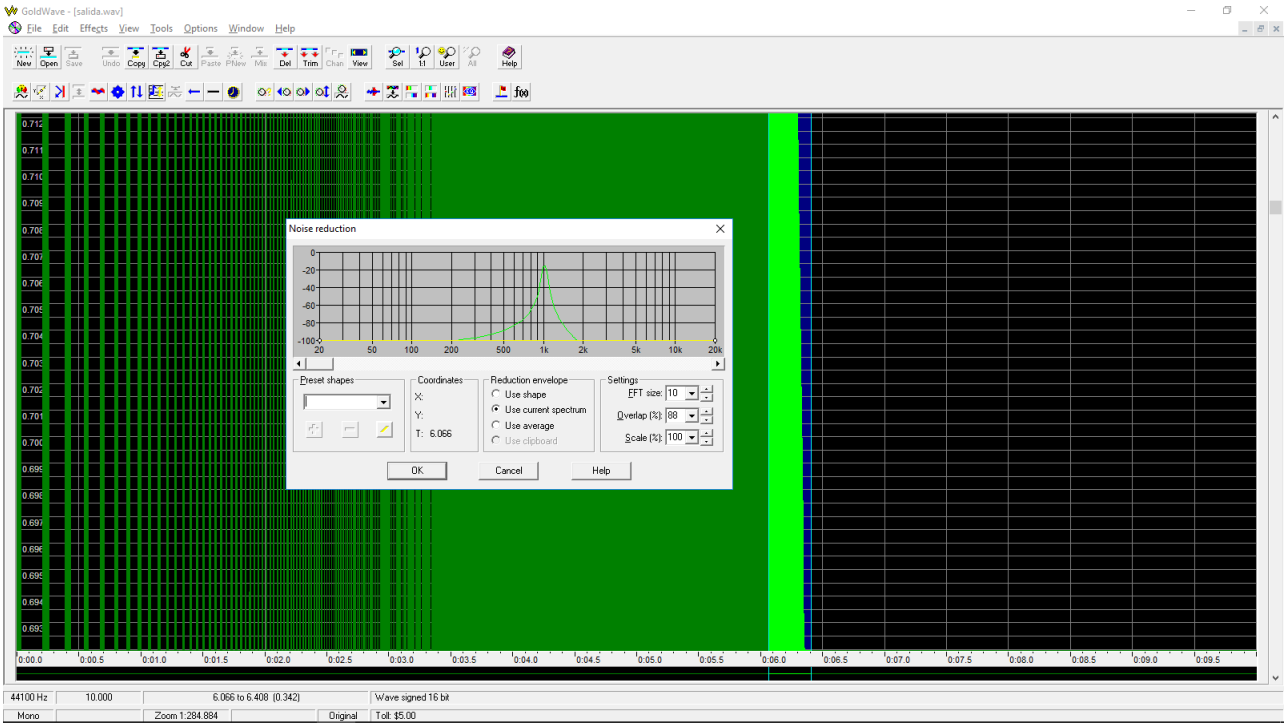

Figura 5: Frecuencia de corte donde la señal decae 3dB

La salida mostrada en la Figura 4, fue obtenida calculando solo 20 coeficientes para el filtro. A continuación se anexan dos Figuras más, en donde se usan 30 y 40 coeficientes para el filtro respectivamente.

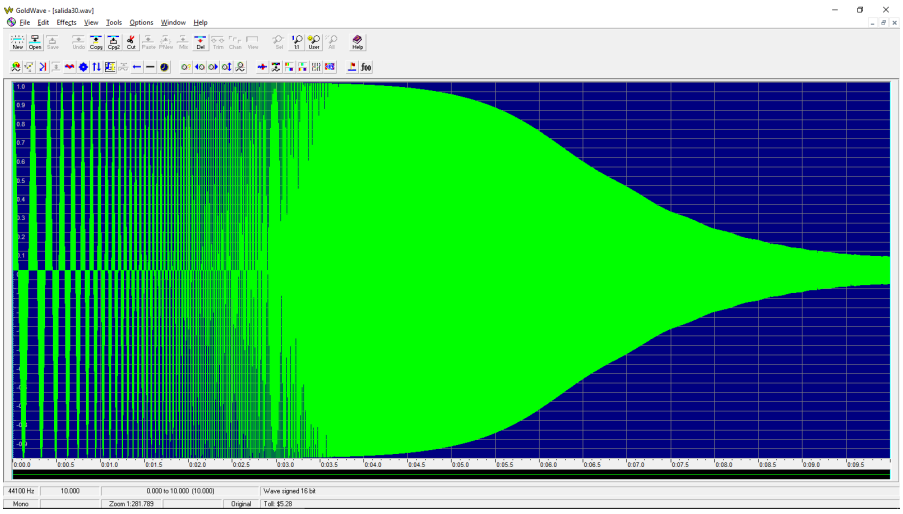

Figura 6: Salida 1 - 30 coeficientes en el filtro

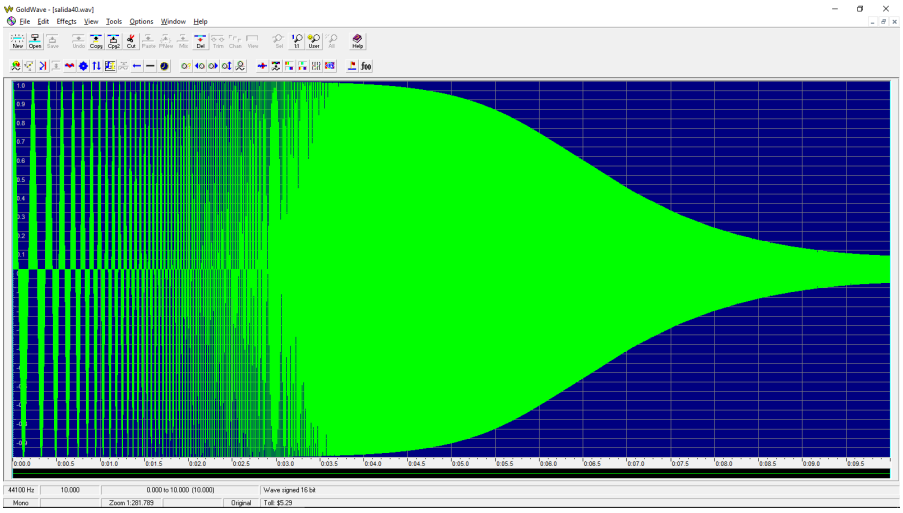

Figura 7: Salida 1 - 40 coeficientes en el filtro

Como puede observarse en las Figuras 6 y 7, mientras más coeficientes son usados para representar el filtro, este se comporta más como un filtro ideal., debido a que al momento de realizarse la convolución, la señal del filtro cuenta con más puntos, permitiendo así, aumentar la precisión de la convolución.

4. Conclusiones

Los filtros pasabajas tienen una gran cantidad de aplicaciones posibles, filtrado de ruido, en comunicaciones, electrónica, etc. Debido a que permiten el paso de una banda de frecuencias menores a la frecuencia de corte del filtro. Las frecuencias mayores a la misma son filtradas (atenuadas). Por lo cual, son una herramienta importante en el trabajo con comunicaciones y señales, donde muchas veces resulta de interés, el solo obtener cierto rango de frecuencias de una grabación o una transmisión.
Por ejemplo, cuando se realiza una grabación de audio mediante un micrófono en una habitación con una lampara eléctrica, la cual podría generar "ruido" en la grabación al estar encendida. Por lo cual, resultaría útil filtrar el espectro de frecuencias producido por la lampara de la grabación.
La implementación más sencilla de un filtro pasabajas es mediante un circuito RC, en el cual, la frecuencia de corte del filtro estará dada por los valores de la resistencia y el capacitor, en aplicaciones de electrónica primero se propondría un valor para el capacitor y posteriormente el de la resistencia para así obtener la frecuencia de corte deseada.
Sin embargo, con el avance de la tecnología ahora resulta mucho más sencillo el simular un circuito RC, ya que puede ser hecho mediante software en pocas lineas de código mediante el uso de la convolución.

Referencias

[1] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 2011.

[2] Sengpielaudio, "Rc filter and cutoff frecuency [online]. disponible en: http://www.sengpielaudio.com/calculator-rcpad.htm."