Proyecto
GrAVFI

www.gravfi.org.ar

Taller de Herramientas en GNU Octave para Teoría de las Comunicaciones

Herramientas de simulación para trabajar en tiempo y frecuencia.

GNU Octave

Octave esta bajo licencia GPL.
www.octave.org

Software Libre (Licencia GPL)

Software libre significa entre otras cosas que las licencias de los programas no requieren de un pago para su uso ni para su distribución. Ello es posible gracias a la GPL. GPL son las iniciales de General Public License, Licencia Pública General cuyo nombre completo es GNU GPL. La GPL permite que cualquiera pueda extender, adaptar o modificar los programas cubiertos por la licencia. También permite que distribuya los resultados de su trabajo y que incluso cobre por ello. La GPL obliga a que el código fuente de un programa esté disponible con la distribución de las copias de ese programa. De este modo, el usuario puede modificar el programa para adaptarlo a sus necesidades y puede vender o distribuir copias de sus modificaciones, siempre y cuando el programa modificado esté también bajo la GPL e incluya los códigos fuente.

La GNU/GPL te obliga a dar a los otros lo que has recibido. El conocimiento que te ha beneficiado y las aportaciones que has hecho a ese conocimiento han de estar también disponibles para otros. La GPL te prohíbe transformar el software libre en software propietario: toda modificación de software GPL ha de continuar siendo GPL.

La GNU/GPL no significa que no haya derechos de autor, que no se reconozcan las autorías de los programas o las modificaciones a los programas. Todo lo contrario. Todas las aportaciones son protegidas por los derechos de autor. Pero los autores aceptan que su trabajo se vea modificado y adaptado por otros, siempre y cuando esas modificaciones sigan siendo libres y estén disponibles con los códigos fuentes, para que puedan a su vez ser modificadas.

Dado que los autores de los programas los ponen a disposición de cualquiera libres de cargas, los programas bajo la GNU/GPL no tienen GARANTÍAS. Se ofrecen "tal cual" y lo que suceda con ellos cae dentro de los riesgos que asumas. Cada uno es responsable de los programas GNU/GPL que decide instalar y probar.

ESTE MANUAL SE SOMETE A LA GPL. Se puede copiar y modificar y se ofrece cumpliendo lo establecido en la GNU/GPL, por lo que sus copias y modificaciones tienen que seguir estando disponibles y ser "copy left" (permitido copiar).

Igualmente, los autores admitimos y agradeceremos toda clase de sugerencias para mejorar el manual, pero no damos garantías de el... ;-)

En GNU/Linux lo importante es aprender. Y aprender es un acto de voluntad. Sin él, no hay manual ni apuntes que sirvan para nada.

Referencias:

- GNU GPL Versión 2, junio de 1991. Versión oficial en inglés. Url: http://www.gnu.org/copyleft/gpl.html
- Traducción no oficial al español de la GPL por Jesús González Barahona y Pedro de las Heras Quirós. Muy recomendable si no te desenvuelves bien con el inglés. Url: http://es.gnu.org/Licencias/gples.html
- Términos de la licencia GPL y GNU. Interesante. Url: http://www.marquese.net/relatos/relatos/mvarios/linux/gpl.htm
- Nota: Esta explicación de la licencia GPL pertenece a el Grupo de Usuarios de Linux de Canarias http://cila.gulic.org/ (contiene algunas pequeñas modificaciones)

Sobre este manual y el curso

Este curso tiene como propósito orientar a los estudiantes de Teoría de las Comunicaciones en el uso de esta poderosa herramienta. Además tiene la intención de colaborar con la formación de conceptos sobre los temas de la materia. Este manual funciona como guía de un taller para aprender a utilizar estas herramientas.

Este manual fue originalmente creado para Matlab. Matlab es una herramienta poderosísima y la mas usada en el área de ingeniería. Es comercial pero en cierta medida tiene algo de la filosofía del software libre ya que muchas de sus librerías .m son editables y hay grandes comunidades de programadores que intercambian herramientas. La gran diferencia radica en que Matlab es software propietario y si bien su licencia no es de las mas caras, es inaccesible para la mayoría de los interesados en usarlo. (inaccesible legalmente) .

Como todos sabemos en nuestro país como en todos los países en vías de desarrollo la gran mayoría del software que se utiliza es “copiado ilegalmente”. Este “delito” de copiar programas que uno necesita para trabajar y progresar, con el crecimiento del software libre esta pasando a ser una tontería, sobretodo en el área de programación e ingeniería. Cada vez hay mas y mejores herramientas en GPL y de empezar a usarlas y a aportar a la comunidad de software libre cada vez va a haber mas. Seguramente que al principio va a haber muchas dificultades, pero si entendemos hasta que punto vale la pena comenzaremos el esfuerzo de migrar hacia el software libre. Debemos comprender que no solo podemos tener problemas legales por usar software propietario, en realidad ese es el menor problema, el verdadero problema es que a medida que mas usamos software que no podemos pagar ni modificar ni adaptar ***mas dependientes nos volvemos como universidad y como país*** limitando drásticamente las posibilidades de desarrollarnos.

Entonces es bastante ilógico trabajar de esta manera teniendo la posibilidad de utilizar software libre, el cual es libre, modificable y en muchas aplicaciones de mejor calidad . La dificultad en el uso de Software libre, esta en que no es muy conocido lo cual asusta a usuarios novatos. Es por esto que en este curso se intentara orientar y ayudar a la instalación y uso de software libre.

Tenemos la verdadera intención de que este manual continúe creciendo y de

seguir desarrollando toolboxes didácticos para la materia. Para esto queremos unir mas estudiantes al grupo que hemos formado y de esta manera continuar con el trabajo.

Como próximo proyecto esta crear un toolbox con toda la que hace a las comunicaciones digitales que se ven en la materia.

Introducción a GNU Octave

Nota: Las explicaciones que prosiguen son limitadas a el uso que se dará de cada herramienta. Tener siempre en cuenta la posibilidad de hacer un help para obtener mas información sobre como funciona cada herramienta. Las .m mencionadas están en cualquier octave reciente o en su defecto en toolbox OCST . También se utilizan algunas .m de Octave_forge .

Definición de variables

Asignamos valores numéricos a las variables simplemente tecleando las expresiones Correspondientes

a = 1+2
lo cual resulta:
a =3
Si colocamos ; al final de la expresión, el resultado se almacena en a pero no aparece en pantalla.
Por ejemplo teclear
a = 1+2;

Octave al igual que Matlab utiliza los siguientes operadores aritméticos :

+ suma
- resta
* multiplicación
/ división
^ potencia
' transposición

Una variable puede ser asignada mediante una fórmula que emplee operadores aritméticos, números o variables previamente definidas. Por ejemplo, como a estaba definida de antemano,
la siguiente expresión es válida:

b = 2*a;

Para visualizar o recordar el valor de una variable que ha sido previamente asignada basta con teclearla de nuevo. Si tecleamos:

b
obtenemos:

b =6

Si la expresión no cabe en una línea de la pantalla, podemos utilizar una elipsis, esto es, tres o mas puntos suspensivos:

```
c = 1+2+3+...
5+6+7;
```

Existen algunas variables predefinidas en Octave por ejemplo:
i - sqrt(–1)
j - sqrt(–1)
pi - 3.1416...
o sea que, si introducimos:

```
y= 2*(1+4*j)
```

tendremos
y=2.0000 + 8.0000i
Existe también una serie de funciones predefinidas que pueden ser empleadas para asignar valores a nuevas variables, por ejemplo:

abs: valor absoluto de un número real o módulo de un número complejo
angle: fase de un número complejo en radianes
cos: función coseno, con el argumento en radianes
sin: función seno, con el argumento en radianes
exp: función exponencial

Por ejemplo, con la y definida anteriormente:

```
c = abs(y)
```

resulta en

c =8.2462

Mientras que:

```
c = angle(y)
```

resulta en
c =1.3258

y con a=3 como definimos antes:

```
c = cos(a)
```

resulta en
c =-0.9900

Mientras que:

c = exp(a)

resulta en

c =20.0855

Nótese que exp puede usarse con números complejos. Por ejemplo, para el

y = 2+8i definido anteriormente:

c = exp(y)

resulta en

c =-1.0751 + 7.3104i

Definición de matrices

Octave está basado en el álgebra de vectores y matrices, incluso los escalares son considerados matrices de elementos. Así que las operaciones entre vectores y matrices son tan simples como las operaciones de cálculo comunes que ya hemos revisado. Los vectores pueden definirse se dos formas. Por una lado, podemos definirlos explícitamente, introduciendo los valores de los elementos:

v = [1 3 5 7];

este comando crea un vector de dimensiones con los elementos 1, 3, 5 y 7. Podemos usar

comas para separar los elementos. Además, podemos añadir elementos al vector, teclear:

v(5) = 8;

resulta en el vector v = [1 3 5 7 8]. Los vectores definidos con anterioridad pueden

servirnos para definir nuevos vectores, por ejemplo:

a = [9 10];

b = [v a];

origina el vector b = [1 3 5 7 8 9 10].

El otro método se utiliza para definir vectores con elementos equi-espaciados:

t = 0:.1:10;

origina un vector de dimensiones con los elementos 0, .1, .2, .3,...,10.

Nótese que en la definición de t, el número que aparece en medio define el incremento de un elemento al siguiente. Si sólo tenemos dos números, el incremento por defecto es 1. Así:

k = 0:10;

da lugar a un vector de dimensiones 1x11 con los elementos 0, 1, 2, ..., 10.

Para definir una matriz A=[1 2 3;4 8 2;2 3 5] los elementos de las filas separados por espacios, y las columnas se separan con punto y coma.

Definición de Funciones

Las funciones se aplican elemento a elemento:

t = 0:10;

x = cos(2*t);

origina un vector x cuyos elementos valen Cos(2t) para t=0,1,2.....10

A veces necesitamos que las operaciones se realicen elemento a elemento. Para ello precedemos el operador correspondiente de un punto ".". Por ejemplo, para obtener un vector x que contenga como elementos los valores de x(t)=t*cos(t) para unos instantes de tiempo determinados, no podemos multiplicar simplemente el vector t por el vector cos(t). Lo que hacemos es:

t = 0:10;

x = t.*cos(t); (nótese el punto después de t produce el producto punto entre los vectores)

Información general

Octave detecta las mayúsculas y minúsculas como diferentes, de modo que "a" y "A" serán dos variables distintas.

Las líneas de comentario en los programas deben estar precedidas de "%"

Mediante el comando **help** podemos obtener ayuda on-line. Si tecleamos help aparecerá todo un menú de temas sobre los que existe la ayuda y si tecleamos help seguido del nombre de una función recibiremos ayuda específica para dicha función.

El número de dígitos con que Octave representa los números en pantalla no está relacionado

con la precisión con que estos han sido calculados. Para cambiar el formato de pantalla teclearemos '**format** short e' si queremos notación científica con 5 cifras significativas, '**format** long e' para notación científica con 15 cifras significativas y '**format** bank' para tener sólo dos dígitos decimales.
Los comandos **who** y **whos** nos dan los nombres de las variables definidas actualmente en el espacio de trabajo (workspace).
.
El comando **length(x)** nos da la longitud del vector x y **size(x)** las dimensiones de la matriz x.

Salvar y recuperar datos desde un fichero
Es muy probable que cuando usemos Octave estemos interesados en guardar los vectores y matrices que hemos creado. Para hacer esto sólo tenemos que hacer: **save** nombre_del_fichero
y para recuperar dichos datos en otra sesión :
load nombre_del_fichero

Para mas información escribir help save.

También suele ser muy útil guardar todo lo que aparece en la pantalla de comandos . Para esto usamos el comando:

diary nombre_del_fichero

Raíces de ecuaciones de grado n (se puede pasar por alto)

Si queremos encontrar las rices de una ecuación por ejemplo: X^2+ 2*X +1

r=**roots**([1 2 1]) ------ Los coeficientes se ponen en orden decreciente
Octave entregara el resultado:

```
r =
   -1
   -1
```

donde r será un vector de 2x1 cuyos elementos son las raíces del polinomio con coeficientes [1 2 1]

Si queremos encontrar un polinomio a partir de sus raíces por ejemplo tenemos las rices [-1,-1]
Escribimos;
p=**poly**([-1 –1])

```
p =
    1    2    1
```

donde p será un vector 1x3 cuyos valores serán los coeficientes en orden decreciente del polinomio cuyas rices son [-1 –1].

Nótese que **roots** y **poly** son funciones inversas.

Si quiero **resolver un sistema de ecuaciones** definido por las matrices Ax=B puedo despejar su resultado simplemente escribiendo

x=A\B ;

Esto computará la inversa de la matriz (A) y hará el producto por la matriz B, lo que resolverá el sistema de ecuaciones.

Plot (para graficar señales)

La función Plot se utiliza de la siguiente manera:

plot(t,y) ; % Grafica los puntos dados por los pares ordenados (t(1),y(1)) (t(2),y(2))....(t(n),y(n)) y une estos puntos.

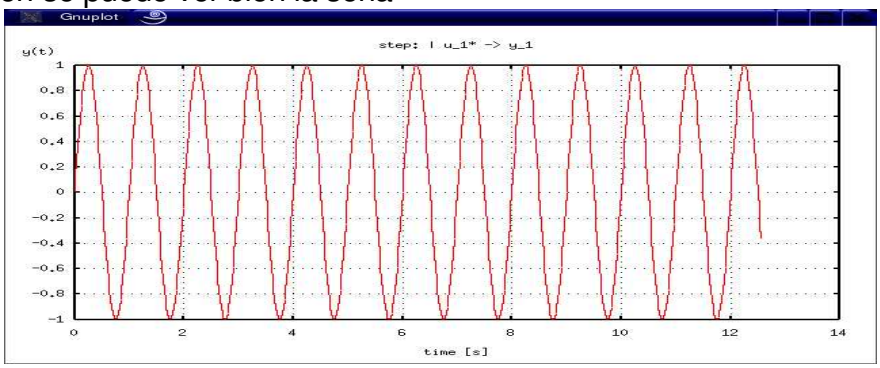
Por ejemplo si quiero graficar la función sin(2*pi*t) entre 0 y 4*pi :

Genero un vector equiespaciado de base de tiempo con un paso de T=1/(50*f) siendo f la frecuencia de la señal . Esto se hace así para lograr una buena visualización aunque con un paso mas largo también se puede ver bien la seña

t=0:1/50:4*pi ;

y=sin(2*pi*t);

plot(t,y);



Transformada Rápida de Fourier

En el uso de esta función se centra el mayor conflicto a la hora de trabajar. Nosotros en esta primera parte del taller lo que buscamos es simular modulaciones y detecciones analógicas. Queremos ver su comportamiento temporal y espectral, pero nuestro gran problema es que estamos trabajando con elementos discretos, que de alguna debemos lograr que simulen a los continuos. Uno de nuestros principales objetivos fue que este Taller sea absolutamente didáctico, por lo que se complico el hecho de mostrar en pantalla los espectros en forma ideal.“““““

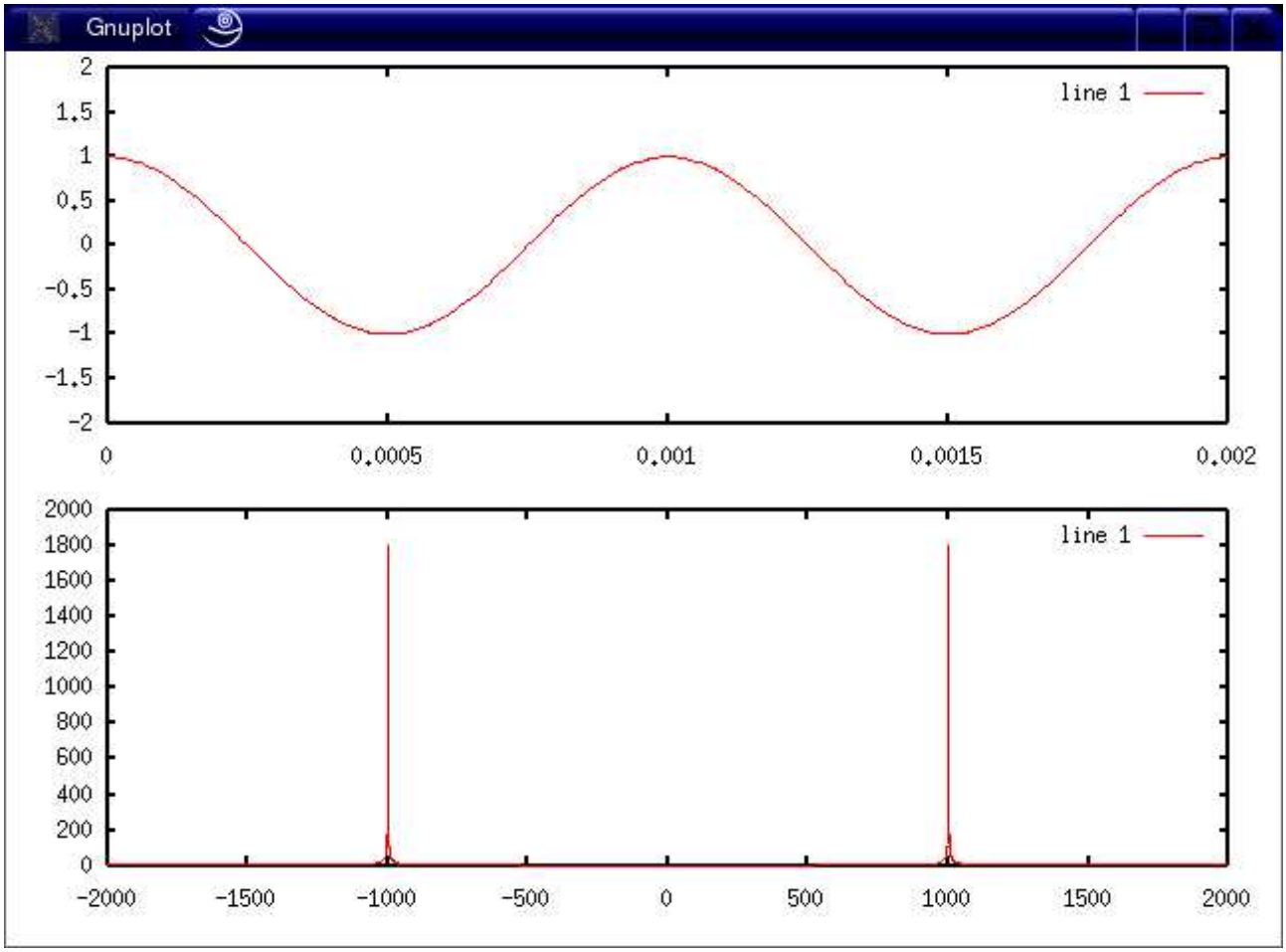
Nos costó mucho trabajo lograr una buena visualización de los espectros, y pasamos por varias semanas de prueba y error, y debemos admitir alguna carencia de conceptos teóricos sobre señales digitales, que al final del trabajo disminuyó. Con este manual se adjunta un apunte (en ingles) que explica muy bien todos los conceptos teóricos que dan a entender el por que de algunos trucos que se hacen para la correcta visualización del espectro.

Pero empecemos con un ejemplo.

Coseno visualizado en Tiempo y Frecuencia .

Vamos a graficar un coseno de frecuencia de 1KHz en tiempo y en frecuencia. Con este simple ejemplo veremos la técnica para lograr una buena visualización del espectro y de la señal en el tiempo.

```
t=-0:0.00001:1 ;          %vector para ser usado como base de tiempo (mayor precisión)

fs=4000 ;                 %frecuencia de sampleo que utilizaremos para poder visualizar bien
el espectro

x=0:1/fs:1 ;              %vector de base de tiempo para usar en la transformada rápida de
fourier
y=cos(2*pi*1e3*t) ;%vector con los valores del coseno para ser graficados en tiempo

yfft=cos(2*pi*1e3*x) ;  %vector con los valores del coseno para ser transformados

YFFT=abs(fft(yfft)) ;    %vector transformada de fourier del coseno

n=length(YFFT) ;

w=(-0.5*(n-1):0.5*(n-1))'  *fs/n ;    % vector base de frecuencia para centrar el espectro

subplot(2,1,2) ; plot(w,YFFT) ;    %graficamos el espectro

subplot(2,1,1) ;  axis([0 2e-3 -1 1]) ; plot(t,y) ; axis ;  %graficamos la señal en el tiempo
```



Aclaracion.

En el ejemplo anterior usamos la función FFT . Como nos interesa graficar el modulo de la transformada le aplicamos el ABS . Luego generamos un vector w, que se usa como base de frecuencias para la grafica del espectro. Este vector w esta creado teniendo en cuenta criterios convenientes para simular que el vector creado por FFT corresponde a la transformada de fourier de un sistema continuo . Se puede apreciar que se divide la cantidad de elementos del vector en 2 porciones una positiva y otra negativa y se la relaciona con la frecuencia de sampleo , ya que FFT separa los picos del espectro que genera de acuerdo a esta frecuencia.

Como practica para entender mejor lo anterior , proponemos como ejercicio hacer la FFT del vector creado con el proposito de ser ploteado en el tiempo. Ahi veremos las dificultades de que los vectores tengan esas caracteristicas.

En este ejemplo y en los que siguen se usan funciones o prametros de funciones que no han sido explicados en la introducción. Si quedan dudas con respecto a estos casos, utilizar siempre el comando HELP . Durante el taller se explicaran estos casos.

Toolbox de Comunicaciones Didáctico

Hemos creado una serie de funciones para octave que simplifican los procedimientos de simulacion.

Es importante comprender los programas paso a paso, y buscar el fundamento teorico detras de los pasos seguidos. Recordamos leer el apunte sobre transformada discreta de fourier.

coseno.m

```
function [t,y] = coseno(frec,amp)

%[t,y] = coseno(frec,amp)

% genera un coseno de frecuencua frec Hz y amplitud amp . genera un vector tiempo

%correspondiente a 4 ciclos de la señal


t=0:1/(50*frec):10/frec ;

y=amp*cos(2*pi*frec*t) ;
```

cosfft.m

```
function [t,y] = cosfft(frec,amp)

%

%[t,y] = cosfft(frec,amp)

%
```

```
% genera un coseno de frecuencua frec Hz y amplitud amp . genera un vector tiempo
%correspondiente a 4 ciclos de la señal
% este tipo de coseno tiene la ventaja que es conveniente para ser transformado con fft
%para ser bien visto cuando se plotea
%Es bueno tomar muchos ciclos pero con frecuencia de sampleo baja.
t=0:1/(4.095*frec):1000/frec ;  %para que FFT trabaje mejor es conveniente que el
                                %vector tenga 2^n elementos .
y=amp*cos(2*pi*frec*t) ;
```

fastft.m

```
function [w,Y] = fastft(y,f)

%transforma usando fft pero entrega el modulo de la transformada y el vector w .
%para ser usada con cosfft anda de lux
%f tiene que ser la frecuencia maxima de la señal
n=length(y) ;
w=(-0.5*(n-1):0.5*(n-1))'  *4*f/n ;
Y=abs(fft(y));
```

Representación espectral y temporal de un señal de AM

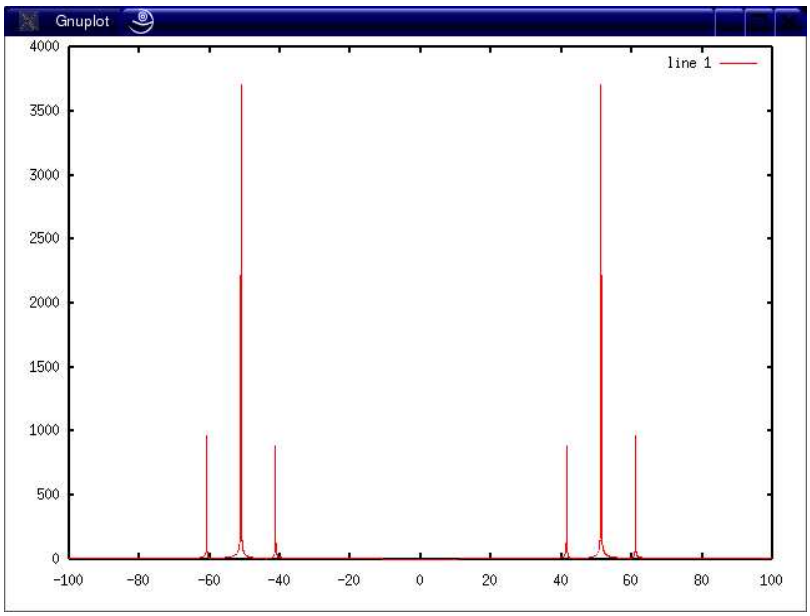
Para hacer un ejemplo n didáctico podemos hacer una banda base de 10 Hz modulada con una portadora de 50 Hz . Ambas cosenoidales . Usaremos un indice de modulación de 0.5

De aquí en mas haremos un procedimiento para obtener graficas temporales y otro para obtener graficas espectrales.

Para obtener el espectro de esta modulación usaremos las funciones creadas anteriormente.

```
[t,port]=cosfft(50,1) ;
am=(2+cos(2*pi*10*t)).*port
;
[w,Y]=fastft(am,50);
plot(w,Y) ;
```

Aquí se puede ver el espectro de la señal de AM .



Para crear la respuesta temporal creamos la siguiente función.

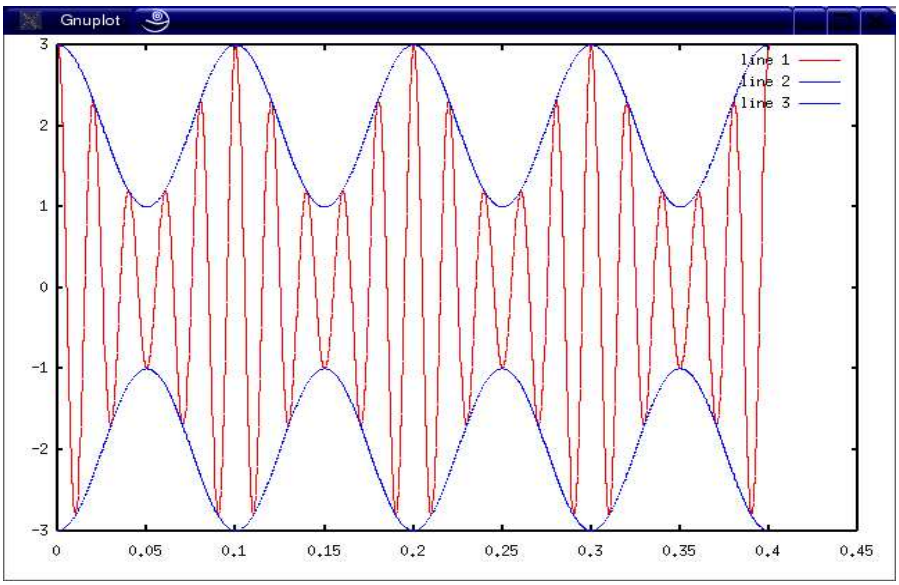
am.m

```
function      [AM,t] = am(Em,fm,Ec,fc);
%
% [AM,t] = am(Em,fm,Ec,fc)
% Crea una señal de AM
% fm : Frecuencia de la Modulante
% fc : Frecuencia de la Portadora
% Em : Amplitud de la señal modulante
% Ec : Ampltud de la Portadora
%da como resultado AM=(Ec+Em*cos(wm*t))*cos(wc*t)
%y grafica la señal con sus envolventes



t=0:(1/(50*fc)):(4/fm) ; %el espacio de tiempo tiene un paso de Tc/50
AM=(Ec+Em*cos(2*pi*fm*t)).*cos(2*pi*fc*t);
plot(t,AM) ; %grafico 4 periodos de la modulante
hold on ;
plot(t,Em*cos(2*pi*fm*t)+Ec,'  b'  ) ; Grafico las envolventes
plot(t,-Em*cos(2*pi*fm*t)-Ec,'  b'  ) ;
hold off ;
```

Usando la funcion para crear la señal de AM deseada :

[AM,t]=am(1,10,2,50) ;

vemos como la banda base graficada en azul , representa la envolvente de la señal.



Para graficar las respuestas en tiempo y frecuencia de forma mas simple , hemos hecho esta funcion:

amtyf.m

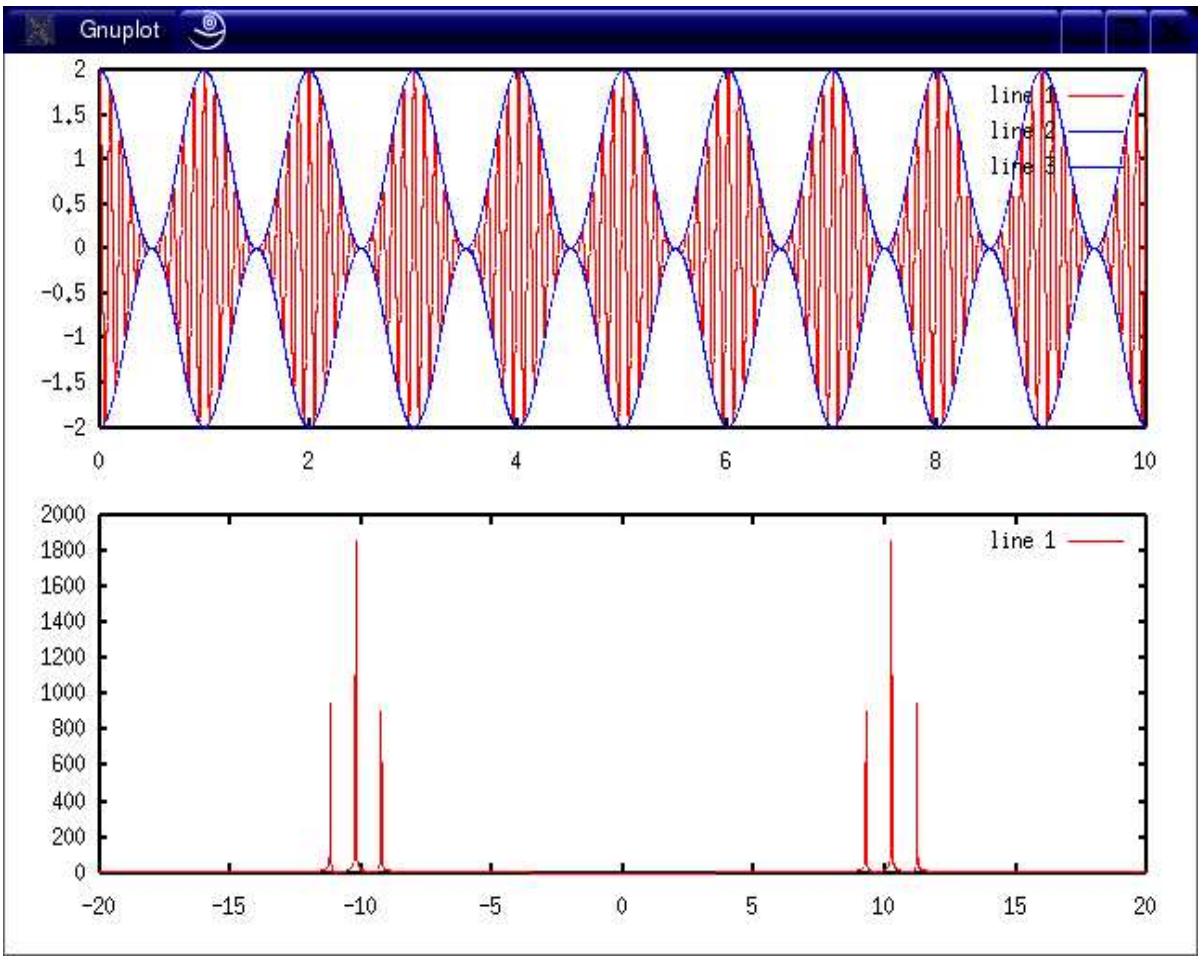
```
function [t,am,w,AM,amft,m,portt,portf] = amtyf(Em,fm,Ec,fc,plote)
%
% AM en tiempo y Frecuencia
%[t,am,w,AM,m,portt,portf] = amtyf(Em,fm,Ec,fc,plot)
%
%ENTRADAS
% fm : Frecuencia de la Modulante
% fc : Frecuencia de la Portadora
% Em : Amplitud de la señal modulante
% Ec : Ampltud de la Portadora
%plote=1 plotea resultado en froma linda
%
%SALIDAS
%t:  es un vector tiempo para graficar resp temporal
%am: es la señal de am en el tiempo
%w : es el vector frecuencias
%AM: es el espectro de la señal de am
%m : es el indice de modulacion
%portt: señal portadora para respuesta temporal
%portf: señal portadora para respuesta en frecuencia
%
%NOTA
%
%La frecuancia de sampleo de am (señal temporal) es de 50*fc
%es decir que el vector base de tiempo es t=0:(1/(50*fc)):(4/fm)

%RESPUESTA ESPECTRAL
```

```
[tft,portf]=cosfft(fc,1) ;
amft=(Ec+Em*cos(2*pi*fm*tft)).*portf ;
[w,AM]=fastft(amft,fc);

%RESPUESTA TEMPORAL

t=0:(1/(50*fc)):(10/fm) ; %el espacio de tiempo tiene un paso de Tc/50 y tiene 10 ciclos de la señal de am
portt=cos(2*pi*fc*t);
am=(Ec+Em*cos(2*pi*fm*t)).*portt;
m=Em/Ec ;%indice de modulacion

if plote==1

  subplot(2,1,1);clearplot;plot(t,am) ;              %grafico
              hold on ;
              plot(t,Em*cos(2*pi*fm*t)+Ec,'  b'  ) ;
              plot(t,-Em*cos(2*pi*fm*t)-Ec,'  b'  ) ;
  hold off ;


              subplot(2,1,2);clearplot; plot(w,AM) ;
              hold off ;
              subplot(111) ;% devuelvo a la normalidad el handle de la figura de ploteo

end;
endfunction
```

Podemos probar esta funcion con la misma señal del ejemplo anterior con la diferencia que usaremos un **indice de modulacion mayor que 1, y veremos la distorsion que esto produce en la envolvente. Tambien observaremos el caso limite cuando el indice de modulacion es igual a 1 .**

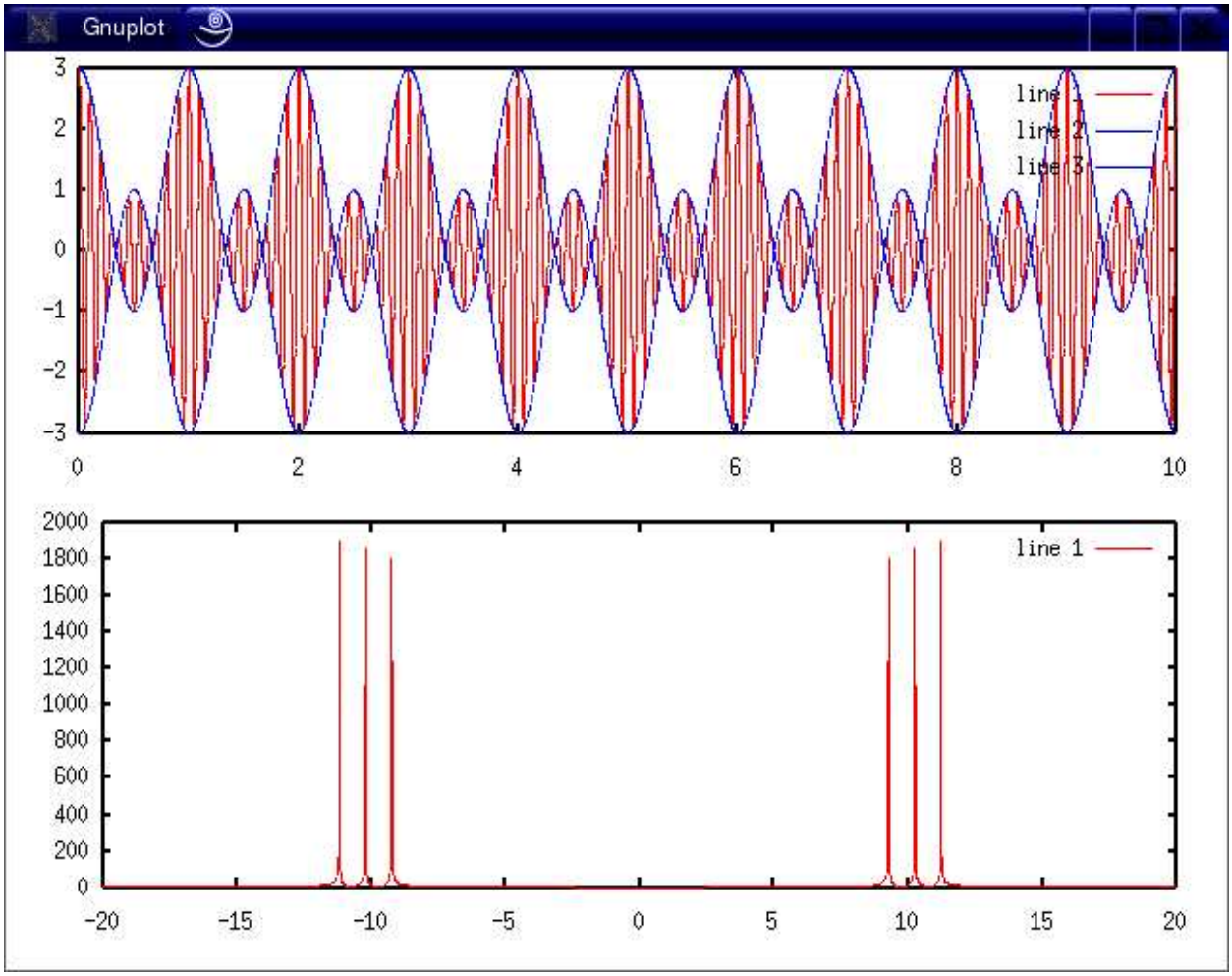
Si el indice de modulacion es igual a 1

```
amtyf(1,1,1,10,1);
```



SI el indice de modulacion es mayor que 1 .

amtyf(2,1,1,10,1); %indice de modulacion igual a 2 .



En la grafica notamos la distorsion de la envolvente, razon por la cual no se podria recuperar la banda base en forma adecuada.

Señal de AM con banda base no-cosenoidal

Octave puede interpretar strings como comandos. Utilizando**strrep y eval** (ver su help) hemos creado una funcion que interpreta bandas base en funcion de t, y plotea la señal de AM en tiempo y frecuencia.

amtyfx.m

```
function [ti,am,w,AM] = amtyfx(mod,fm,Ec,fc,plote)

%[t,am,w,AM] = amtyfx(mod,fm,Ec,fc,plot)
%
%ENTRADAS
```

```
% mod : string en funcion de t que representa la banda base
% fm : Frecuencia de la Modulante (ancho de banda base)
% Ec : Ampltud de la Portadora
% fc : Frecuencia de la Portadora
%plot=1 plotea resultado en froma agradable
%
%SALIDAS
%t es un vector tiempo para graficar resp temporal
%am es la señal de am en el tiempo
%w es el vector frecuencias
%AM es el espectro de la señal de am


%Respuesta en Frecuencia

[tft,port]=cosfft(fc,1) ;
moda=strrep(mod,'  t'  ,'  tft'  ) ;
modi=eval(moda,tft) ;
amft=(Ec+modi).*port ;
[w,AM]=fastft(amft,fc);

% Respuesta temporal

ti=0:(1/(50*fc)):(4/fm); %el espacio de tiempo tiene un paso de Tc/50
moda=strrep(mod,'  t'  ,'  ti'  ) ;
modi=eval(moda,ti) ;
am=(Ec+modi).*cos(2*pi*fc*ti);

%PLOTEO

if plote==1
   subplot(2,1,1);clearplot;plot(ti,am) ;              %grafico 4 periodos de la señal de am
                hold on ;
                plot(ti,modi+Ec,'  b'  ) ;
                plot(ti,-1*modi-Ec,'  b'  ) ;
   hold off ;


                subplot(2,1,2);clearplot; plot(w,AM) ;
hold off;
end

endfunction
```

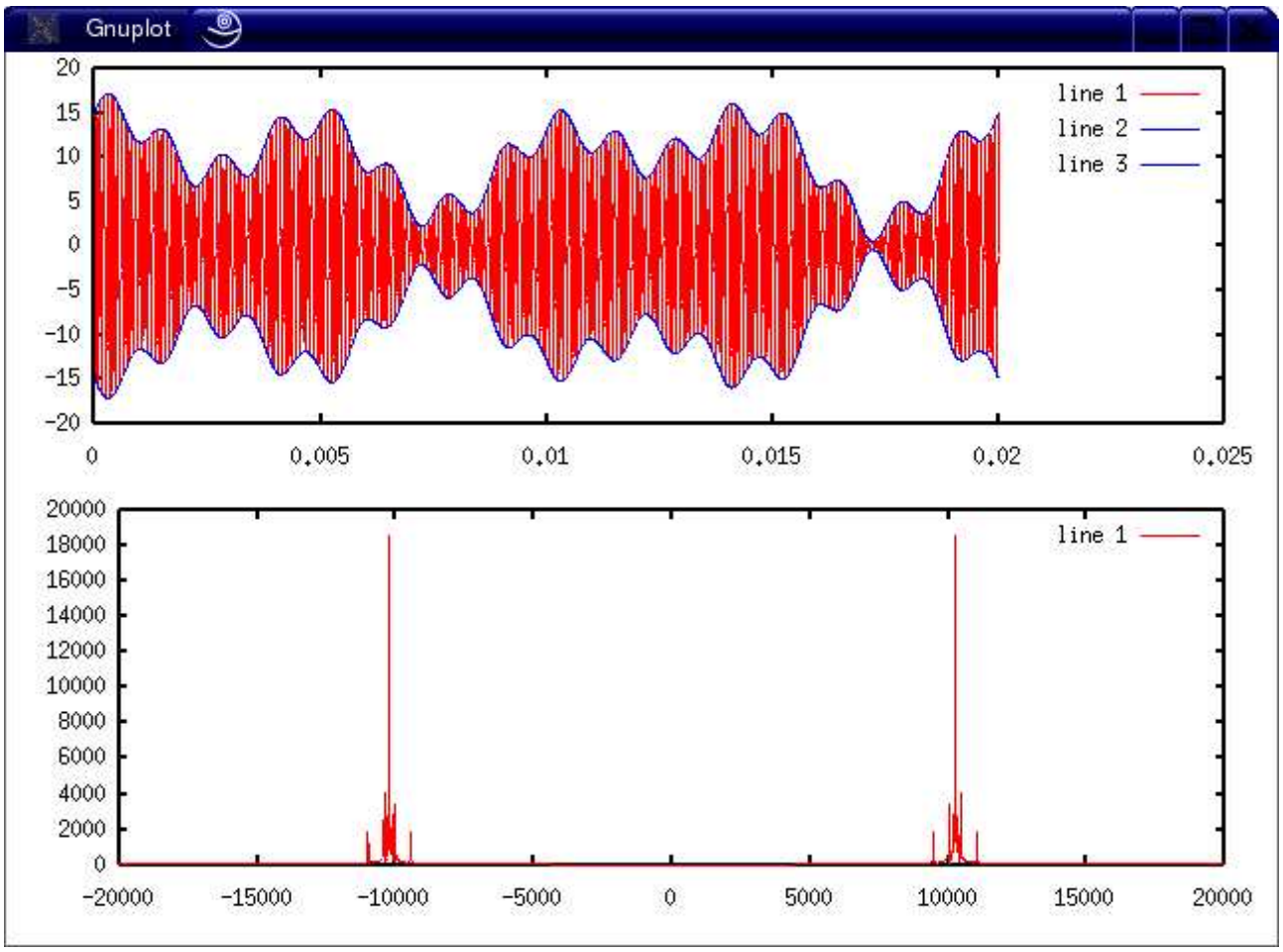
Veamos un ejemplo de su utilización. Supongamos una banda base formada por componentes senoidales de distintas frecuencias como :

```
bbase="3*sin(2*pi*100*t)+4*cos(2*pi*200*t)+2*sin(2*pi*800*t)+cos(2*pi*150*t)"
```

usando la funcion amtyfx.m obtenemos :

amtyfx(bbase,200,10,10000,1); % Usamos una portadora de 10KHz

Como podemos observar, la banda base siempre envuelve a la portadora.



Si tenemos una banda base rectangular:

bbase = square(2*pi*100*t) ;

amtyfx(bbase,100,3,1000,1);

obtendremos