

I.8 Trabajos Prácticos

Ejercicio 1

🖨️ Genere y grafique las siguientes señales

1. senoidal

$$x(t) = A \sin(\omega t + \varphi); \omega = 2\pi f$$

octave/seno.m

```
function x = seno (t,Am,fr,fi)
    n=length(t);
    x=zeros(1,n);
    for k=1:n
        x(k)=Am*sin((2*pi*fr)*t(k)+fi);
    end
endfunction
```

para usarlo:

```
> t=0:pi/12:pi-pi/12;
> x=seno(t,0.5,1,0);
> plot(t,x);
```

2. sinc

$$x(t) = \begin{cases} \frac{\sin x}{x} & x \neq 0 \\ 1 & x = 0 \end{cases}$$

octave/sinc.m

```
function x = sinc (t)
    n=length(t);
    x=zeros(1,n);
    for k=1:n
        if t(k)==0,
            x(k)=1;
        else
            x(k)=sin(t(k))/t(k);
        endif
    end
endfunction
```

para usarlo:

```
> T=pi/12;
> t=-4*pi:T:4*pi-T;
> x=sinc(t);
> plot(t,x);
```

3. onda cuadrada

$$x(t) = \begin{cases} 1, & t/T - \lfloor t/T \rfloor < c \\ -1, & t/T - \lfloor t/T \rfloor \geq c \end{cases},$$

donde c = ciclo de trabajo, $0 < c < 1$, y T = período.

octave/cuad.m

```
function x = cuad (t,pe,ct)
    n = length(t);
    u = t / pe;
    x = ones(1,n);
    for k=1:n
        if (u(k) - floor(u(k))) >= ct,
            x(k) = -1;
        endif
    end
endfunction
```

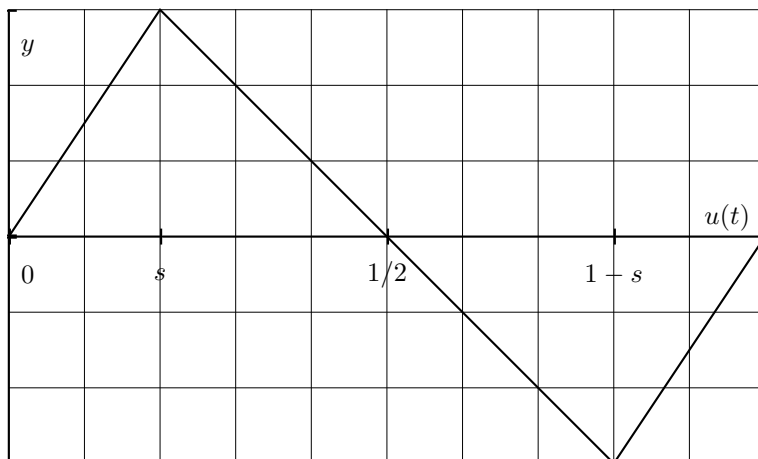
para usarlo:

```
> t=0:1/10:3-1/10;
> x=cuad(t,2,0.3);
> plot(t,x);
```

4. onda triangular

$$x(t) = y(u(t)) = \begin{cases} \frac{u}{s}, & u \leq s \\ \frac{2u-1}{2s-1}, & s < u < 1-s \\ \frac{u-1}{s}, & u \geq 1-s \end{cases}$$

donde $u = u(t) = \frac{t}{T} - \left\lfloor \frac{t}{T} \right\rfloor$, y $0 \leq s \leq \frac{1}{2}$



octave/triang.m

```
function x = triang (t,pe,s)
    n = length(t);
    u = t / pe;
    x = ones(1,n);
    s/=2;
    for k=1:n
        dist=u(k) - floor(u(k));
        if (dist) < s,
            x(k) = dist/s;
        elseif (dist) > s,
            if (dist) < 1-s,
                x(k) = (1-2*dist)/(1-2*s);
            elseif (dist) > 1-s
                x(k) = (dist-1)/s;
            else
                x(k) = -1;
            endif
        else
            x(k) = 1;
        endif
    end
endfunction
```

para usarlo:

```
> t=0:1/10:3-1/10;
> x=triang(t,2,0.3);
> plot(t,x);
```

5. delta de Dirac

$$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}; n \in \mathbb{Z} \quad \text{ó} \quad \delta(t) = \begin{cases} \infty & t = 0 \\ 0 & t \neq 0 \end{cases}; t \in \mathbb{R}$$

octave/delta.m

```
function x = delta (t)
    n = length(t);
    x = zeros(1,n);
    for k=1:n
        if t(k) == 0,
            x(k) = realmax;
        endif
    end
endfunction
```

6. ruido aleatorio

octave/ruido.m

```
function x = ruido (t)
    x = rand(size(t));
endfunction
```

Ejercicio 2

☞ Realice las siguientes operaciones básicas sobre una señal senoidal: compresión, expansión, inversión, rectificación, cuantización en 8 niveles, traslación.

Si mi señal senoidal viene dada por $x(t) = A \sin(\omega t + \varphi)$, defino $x_{\text{nueva}}(t) = x(\alpha t) = A \sin(\omega \alpha t + \varphi)$. Entonces, si quiero

- comprimir la señal, hago $\alpha > 1$;
- expandir la señal, hago $\alpha < 1$;
- invertirla, hago $\alpha = -1$.

Para rectificar la señal, hago $x_{\text{nueva}}(t) = |x(t)| = A |\sin(\omega t + \varphi)|$.

En Octave, tengo los vectores t y x , donde t =dominio y x =rango. Para:

- comprimir, expandir, invertir la señal:

```
t=alfa*t;
```

- rectificarla:

```
x=abs(x);
```

- cuantizarla en n niveles:

octave/cuantizar.m

```
function y = cuantizar (x, n)
    i=length(x);
    amp=max(x)-min(x);
    dc=min(x)+amp/2;
    y=x-dc;
    y/=amp;
    y(1:i)=(floor((n-1)*y(1:i))+0.5)/(n-1);
    y*=amp;
    y+=dc;
endfunction
```

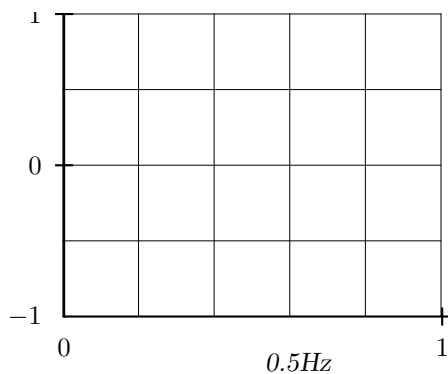
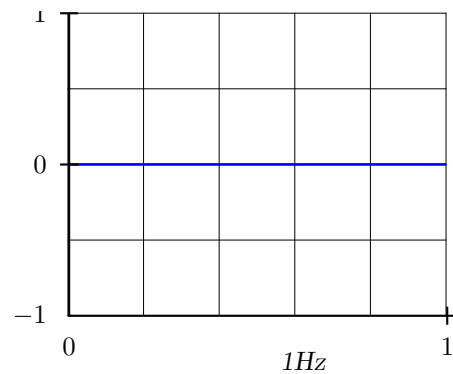
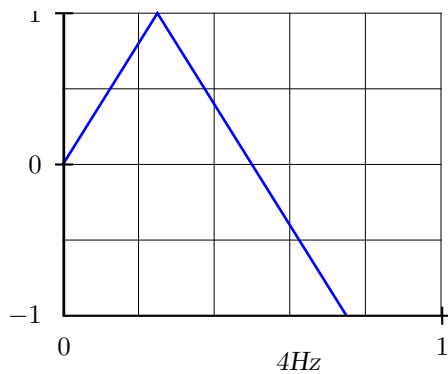
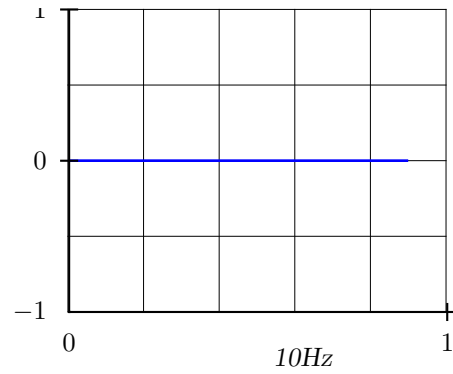
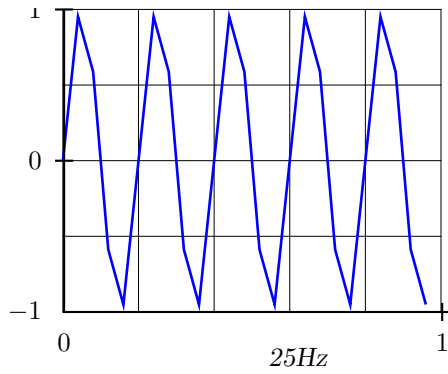
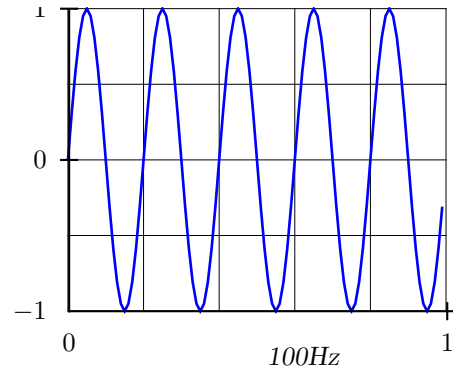
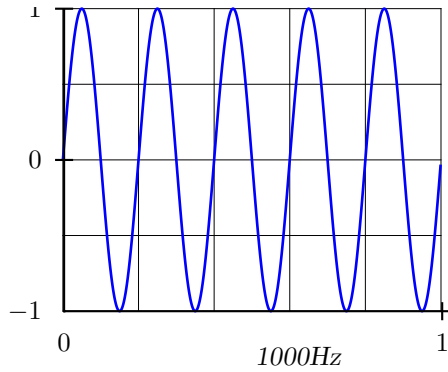
- trasladarla:

```
t=t+d;
```

Ejercicio 3

☞ Discretice una señal senoidal con frecuencia 5Hz y duración 1seg. Utilice las siguientes frecuencias de muestreo: 1000, 100, 25, 10, 4, 2, 1 y 0,5 Hz. Grafique y analice el resultado en cada uno de los casos.

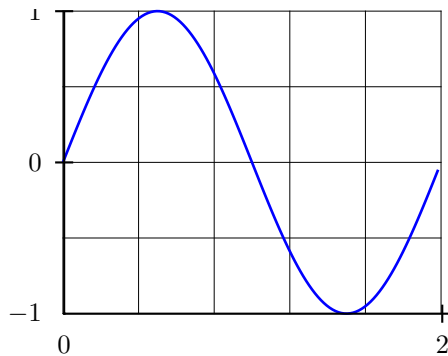
El dominio discreto es $t=0:1/f:1-1/f$; donde f = frecuencia de muestreo. La señal la genero haciendo $x=\sin(2\pi*5*t)$;



Ejercicio 4

☞ Discretice una señal senoidal con frecuencia 4000Hz y duración 2seg., utilizando una frecuencia de muestreo de 129Hz. Grafique el resultado y estime la frecuencia de la onda sinusoidal que se observa en la figura. Analice y obtenga conclusiones.

```
t=0:1/129:1-1/129;
x=seno(t,1,4000,0);
plot(t,x);
```



La onda original tenía 4000Hz, sin embargo, en la figura se ve una onda de 0.5Hz. Esto se debe a que mi frecuencia de muestreo no es lo suficientemente alta, por lo que se produce el fenómeno que apreciamos aquí: el *aliasing*. Esto es porque una onda se confunde, o es un alias, con la original. Si yo no supiera de la onda original, ni me enteraría que esto pasó.

Ejercicio 5

☞ Discretice una señal arbitraria con frecuencia de muestreo de 10Hz y sobremuestréela mediante distintos tipos de interpoladores a 4 veces la frecuencia de muestreo.

```
t=0:1/10:1-1/10;
x=sin(pi*t.^2)+sin(5*t.^2); #mi señal cualquiera a 10Hz
u=0:1/40:1-1/40;
y=interp(x,1/10,u,1);      #interpolada escalon a 40Hz
y=interp(x,1/10,u,2);      #interpolada lineal a 40Hz
y=interp(x,1/10,u,3);      #interpolada sinc a 40Hz
```

octave/interp.m

```
function y=interp(x,T,t,I)
    m=length(t);
    n=length(x);
    y=zeros(1,length(t));
    for j=1:n
        for k=1:m
            if I==1,
                y(k) = y(k) + x(j)*Iescalon((t(k)-(j-1)*T)/T);
            elseif I==2,
                y(k) = y(k) + x(j)*Ilineal((t(k)-(j-1)*T)/T);
            elseif I==3,
                y(k) = y(k) + x(j)*Isinc((t(k)-(j-1)*T)/T);
            endif
        end
    end
endfunction
```

octave/Iescalon.m

```
function x=Iescalon(t)
    if round(100000*t)>=0 & round(100000*t)<100000,
        x=1;
    else x=0;
    endif
endfunction
```


octave/Ilineal.m


```
function x=Ilineal(t)
    if abs(t) < 1,
        x = 1-abs(t);
    else x = 0;
    endif
endfunction
```

octave/Isinc.m


```
function x=Isinc(t)
    if t == 0,
        x=1;
    else
        x=sin(pi*t)/(pi*t);
    endif
endfunction
```

Ejercicio 6

 Genere una señal compleja del tiempo y gráfiquela en 3 dimensiones.

 ¿Qué es una señal compleja del tiempo?

Ejercicio 7

 Genere una señal aleatoria con distribución gaussiana y verifique su ergodicidad.

Como toda señal aleatoria de distribución gaussiana se caracteriza completamente por su media \bar{X} y su varianza σ^2 , me basta con verificar esos parámetros para decidir la estacionariedad y ergodicidad o no de la señal.

octave/ejI7.m

```
function ejI7
# Genero las muestras:
A=randn(100,100);

# Vectores con los params estadisticos
# de cada tiempo (c) y muestra (f):
med_t=zeros(1,100);
var_t=zeros(1,100);
med_m=zeros(1,100);
var_m=zeros(1,100);

# Calculo medias y varianzas entre
# las muestras en cada instante t:
for k=1:100
    med_t(k)=mean(A(1:100,k));
    var_t(k)=var(A(1:100,k));
end


# Ploteo los valores a ver si es estacionaria:
plot(1:100,med_t,";media en el instante t;",
     1:100,var_t,";varianza en el instante t;");

pause

# Calculo medias y varianzas
# a lo largo de cada muestra:
for k=1:100
    med_m(k)=mean(A(k,1:100));
    var_m(k)=var(A(k,1:100));
end

# Ploteo los valores a ver si es ergódica:
plot(1:100,med_m,";media en cada muestra;",
     1:100,var_m,";varianza en cada muestra;");
endfunction
```


Ejercicio 8


 Lea dos señales sonoras desde archivos (p. ej., en formato WAV) y luego súmelas. [...] Guarde el resultado en un archivo del mismo formato y oiga las tres señales.

 en Scilab:

scilab/sumaudio.sce

```
function sumaudio( in1, in2, out)
    // Leo los archivos de audio
    au1=wavread(in1);
    au2=wavread(in2);
    // Los sumo
    if length(au1)==length(au2),
        suma=au1+au2;
    elseif length(au1)>length(au2),
        suma=au1;
        suma(1:length(au2))=suma(1:length(au2))+au2;
    else
        suma=au2;
        suma(1:length(au1))=suma(1:length(au1))+au1;
    end
    // Guardo el archivo nuevo
    wavwrite(suma,44100,out);
endfunction
```

Ejercicio 9


 Lea una señal sonora conocida y súmele un ruido aleatorio. Oiga el resultado y compare con la señal original.

 en Scilab:

scilab/sumaudioruido.sce

```
function sumaudioruido( in, out)
    // Leo el archivo de audio
    au1=wavread(in);
    // Le sumo ruido (rand(a) devuelve una
    // matriz de n aleatorios del tamaño de a):
    suma=au1+rand(au1);
    wavwrite(suma,44100,out);
endfunction
```

Ejercicio 10

 Calcule la SNR y vuelva a ensuciar la señal con 0dB y 100dB...



en Scilab:

- Calcular la relación señal/ruido:

scilab/snr.sce


```
function x= snr( sig, noise)
// Me fijo que tipo son los args:
if typeof(sig) == 'constant' then
    senial=sig;
elseif typeof(sig) == 'string' then
    senial=wavread(sig);
else abort;
end
if typeof(noise) == 'constant' then
    ruido=noise;
elseif typeof(noise) == 'string' then
    ruido=wavread(noise);
else abort;
end
// Calculo las potencias
psenial=sum(senial.^2)/length(senial);
pruido=sum(ruido.^2)/length(ruido);
// Devuelvo la snr
x = 10*log(senial/ruido);
endfunction
```

- Ensuciar la señal:

scilab/ensuciar.sce

```
function x=ensuciar(sig, snr)
// Compruebo de que tipo es sig:
if typeof(sig)=='constant' then
    senial=sig;
elseif typeof(sig) == 'string' then
    senial=wavread(sig);
else abort;
end
// Calculo la potencia de la señal:
psenial = sum(senial.^2)/length(senial);
// Genero el ruido
ruido = rand(senial);
pruido = sum(ruido.^2)/length(ruido);
// Ahora bien, pruido debe ser:
// 10log(ps/pr)=snr => pr=ps/10^(snr/10)
pr = psenial/(10^(snr/10));
// pruido=c*pr => pr=pruido/c
// => pr=pot(ruido/sqrt(c));
c=pruido/pr;
ruido = ruido / sqrt(c);
// Ahora si, tengo la snr requerida:
x = senial + ruido;
endfunction
```

Ejercicio 11


 Utilice una señal sonora conocida y multiplique cada uno de sus elementos por una constante. Oiga el resultado y compare con la señal original.

 en Scilab:

scilab/multiplic.sce

```
function x=multiplic(sig, const)
    // Compruebo de que tipo es sig:
    if typeof(sig)=='constant' then
        senial=sig;
    elseif typeof(sig) == 'string' then
        senial=wavread(sig);
    else abort;
    end
    x=senial*const;
endfunction
```

Ejercicio 12

 Utilice una señal sonora conocida y multiplique cada uno de sus elementos por una recta decreciente que tenga valor 1 en el primer elemento y 0 en el último. Oiga el resultado y compare con la señal original.

 en Scilab:

scilab/multiplicrect.sce

```
function x=multiplicrect(sig)
    // Compruebo de que tipo es sig:
    if typeof(sig)=='constant' then
        senial=sig;
    elseif typeof(sig) == 'string' then
        senial=wavread(sig);
    else abort;
    end
    for k=1:length(senial)
        senial(k)=senial(k).*(1-(k/length(senial)));
    end
    x=senial;
endfunction
```