UNIVERSIDAD NACIONAL DE CORDOBA

Facultad de Ciencias Exactas Físicas y Naturales

Curso:

Herramientas de Calculo y Simulación para Análisis de Redes Eléctricas

Orientado a alumnos de Teoría de Redes

Herramientas a utilizar: MATLAB y CIRCUITMAKER

Objetivos

El objetivo de este curso es orientar a alumnos ,.que estén cursando materias en las que se analicen redes eléctricas, en el uso de herramientas informáticas especificas que ayudan a una mejor comprensión y estudio de la materia. Este curso esta especialmente orientado a los alumnos de Teoría de Redes, pero puede ser de utilidad para alumnos que cursen otras materias con contenidos similares.

La primera parte introduce al alumno en el uso de MATLAB como herramienta de calculo. Aprenderá las bases del funcionamiento de la "Command Window" de MATLAB , y terminara calculando Filtros Modernos de Butterworth y Chebyschev, o calculando residuos en los polos para sintetizar redes. Se orientara al alumno para que aprenda a comprobar los resultados de los ejercicios que realice de la Guía de Trabajos Prácticos, ya sea en calculo de Funciones de Transferencia , Diagramas de Bode o de Nyquist , o gráficos de todo tipo de funciones, Transformadas de Laplace, operaciones matriciales con complejos y variable S, etc.

La segunda parte explica al alumno como utilizar el programa CIRCUITMAKER, que es una **herramienta de simulación recomendada para estudiantes por su simplicidad de uso y su amplia versatilidad**. Inicialmente enseñara como simular redes pasivas, realizar mediciones de voltajes corrientes e impedancias, luego se orientara al alumno para que pueda obtener la respuesta de circuitos RC, RL y RLC en DC de forma tal que podrá verificar los resultados de los ejercicios que realice por anti-transformada de Laplace. También se enseñara como realizar operaciones matemáticas sobre las respuestas de los circuitos. Por ultimo se explicará como obtener la respuesta en frecuencia de redes a través de su simulación, como obtener su diagrama de Bode y se orientara en su interpretación para que el alumno pueda simular filtros de frecuencia y observar su comportamiento de atenuación . En el curso se darán ejemplos sacados de ejercicios de la Guía de Trabajos Prácticos.

INTRODUCCIÓN

Matlab (matrix laboratory) es, conceptualmente, un lenguaje de programación de alto nivel. Contiene amplias librerías de funciones matemáticas que nos permitirán operar con matrices y obtener representaciones gráficas de datos. Iniciamos MATLAB mediante el correspondiente icono en el escritorio de Windows-98 o mediante el ítem correspondiente en el menú de inicio. Acto seguido se abrirá una ventana que nos permitirá ir introduciendo datos y comandos de manera interactiva. Detrás de cada comando debemos pulsar la tecla de retorno ('return', 'enter' o '¿') para que este sea ejecutado.

Para utilizar bajo Linux (tambien bajo windows) , hay una versión gratuita del MATLAB (un programa muy similar) llamado OCTAVE. La forma de trabajo es muy similar.

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;.
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 MATLAB, 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

MATLAB 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.

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)

INFORMACIÓN GENERAL

MATLAB 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 MATLAB representa los números en pantalla no está relacionado con la precisión con que estos han sido calculados. Para cambiar el formato de pantalla tecleareremos '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). Tambien hay un icono para acceder al workspace y observar todas las variables.
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 MATLAB 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
Es mas sencillo hacer esto a través de la ventana de comando file, y elegir la opcion save workspace as y luego para recuperar load workspace.
Para mas información escribir help save.

RAICES DE ECUACIONES DE GRADO N

MATLAB, permite trabajar ya sea utilizando métodos numéricos ó paralelamente se puede trabajar con variables simbólicas.

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

r=**roots**([1 2 1]) ------ Los coeficientes se ponen en orden decreciente
Matlab 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 raices [-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 raices son [-1 –1].

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

Ahora veamos la forma **simbólica** de realizar lo mismo:

Primero hay que declarar las variables a traves de la funcion **syms**

```
syms x
syms a b c real
```

De esta manera se han creado 4 variables: x que será la variable incógnita y a b y c que seran constantes reales (tambien se podrían haber utilizado como constantes complejas). El nombre de las variables puede ser cualquier otro.
Y utilizamos luego la funcion **solve**('expresión matematica',variable a despejar)

```
solve('a*x^2+b*x+c=0',x)
ans =
[ 1/2/a*(-b+(b^2-4*a*c)^(1/2))]
[ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]
```

Nótese que si utilizamos una función y no asignamos una variable al resultado MATLAB siempre guarda el ultimo resultado obtenido en una variable ans
Vemos también que la forma en que **ans** esta escrito de una forma un poco difícil de leer por lo que es muy útil una función que se llama **pretty** hace lo siguiente:

```
pretty(ans)
```

```
[                2                1/2]
[   -b~ + (b~  - 4 a~ c~)   ]
[1/2 ------------------------    ]
[            a~                  ]
[                                ]
[                2           1/2]
[   -b~ - (b~  - 4 a~ c~)   ]
[1/2 ------------------------    ]
[            a~                  ]
```

ahora el resultado es mas legible.

Si luego se desea reemplazar el valor de a, b y c se utiliza la funcion
subs(ecuación,{variable1, variable2, variablen},{valor1,valor2,valorn})

en nuestrocaso
subs(ans,{a,b,c},{1,2,1})

```
ans =
   -1
   -1
```

entrega el valor numérico de las raices.

Hay dos funciones que nos permiten pasar del la forma numérica a la simbólica y viceversa.

Son **sym2poly** (de simbólico a numérico) y **poly2sym** (de numérico a simbólico)

EJ:
sym2poly(x^2+2*x+1)
ans =
1 2 1
poly2sym([1 2 1])
ans =
x^2+2*x+1

SOLUCION DE SISTEMAS DE ECUACIONES

Hay muchísimas formas de resolver un sistema de de ecuaciones. Nombraremos algunas.

Si tenemos una matriz AB=[A B] donde A es la matriz de coeficientes y B es el vector de términos independientes. Podemos reducir la matriz por filas con la funcion **rref**(AB) y asi obtener la solucion del sistema de ecuaciones. A y B pueden contener elementos tanto numéricos como simbólicos.

Otra forma seria con determinantes utilizando la regla de Cramer. La función determinante es **det(A).** Donde A es la matriz cuadrada de coeficientes. Vale la pena repetir que la matriz puede contener elementos simbólicos, y el determinante va a estar en función de esos elementos.

La forma **simbólica** es:

solve('eqn1','eqn2',...,'eqnN','var1,var2,...,varN')

donde eqnn son ecuaciones simbólicas o texto entre comas indicando ecuaciones, y varn son las variables a conocer.

Ejemplo:

[x,y] = **solve**('x^2 + x*y + y = 3','x^2 - 4*x + 3 = 0') entrega

x = [1]

[3]

y = [1]

[-3/2]

el resultado también puede ser simbólico:

[x,y,z]=**solve**('a*x+b*y+c*z=58','b*a*x+c*b*y+c*z=5','c*x+a*y+b*z=0')

x =

-(53*a*c+5*b^2-58*b^2*c)/(-a^2*c+b^2*a*c-b^3*a+c^2*b+c*b*a^2-c^3*b)

y =

-(-5*b*a+58*b^2*a-53*c^2)/(-a^2*c+b^2*a*c-b^3*a+c^2*b+c*b*a^2-c^3*b)

z =

(-5*a^2+5*c*b+58*b*a^2-58*c^2*b)/(-a^2*c+b^2*a*c-b^3*a+c^2*b+c*b*a^2-c^3*b)

y luego si a,b y c fueron declarados como reales (**syms** a b c real) podrán ser reemplazados con la función **subs,** para obtener resultados numéricos

COMANDOS ALGEBRAICOS-LIMITES DERIVADAS E INTEGRALES

En algunos casos puede ser de utilidad si tenemos una expresión simbólica complicada simplificarla, o sacar factor común algún termino. Bueno, esta necesidad también está cubierta por Matlab.

collect(S,v) es una función que rescribe la ecuación simbólica S en términos de la variable v.

Ejemplo:

collect(x^2*y + y*x - x^2 - 2*x,x)

devuelve

(y-1)*x^2+(y-2)

f = -1/4*x*exp(-2*x)+3/16*exp(-2*x)

collect(f,exp(-2*x))

devuelve

(-1/4*x+3/16)*exp(-2*x)

la funcion inversa a collect es:

expand(s)

Ejemplo

expand((y-1)*x^2+(y-2)*x)

ans =

x^2*y+y*x-x^2-2*x

NOTA IMPORTANTE: **Si no se especifica la variable MATLAB elige la letra alfabéticamente mas cercana a la 'x'. Esto funciona así para todas las funciones simbólicas.**

La función para simplificar es:

[R,HOW] = **simple**(S) donde S es una expresión simbólica. Esta función va a probar varios métodos de simplificación, y va a devolver el resultado mas simple que encuentre en la variable R y en la variable HOW guardara un texto con el método utilizado.

Ejemplo

S	R	How
cos(x)^2+sin(x)^2	1	combine(trig)

2*cos(x)^2-sin(x)^2	3*cos(x)^2-1	simplify
cos(x)^2-sin(x)^2	cos(2*x)	combine(trig)
cos(x)+(-sin(x)^2)^(1/2)	cos(x)+i*sin(x)	radsimp
cos(x)+i*sin(x)	exp(i*x)	convert(exp)
(x+1)*x*(x-1)	x^3-x	collect(x)
x^3+3*x^2+3*x+1	(x+1)^3	factor
cos(3*acos(x))	4*x^3-3*x	expand

syms x y positive

log(x) + log(y)	log(x*y)	combine

Matlab tiene funciones especificas para calcular **Limites, Derivadas e Integrales.**

Limites:

limit(F,x,a) limite de la expresión simbólica F cuando **x -> a**

limit(F,x,a,'right') o **limit**(F,x,a,'left') especifica si es limite por la derecha o por la izquierda.

limit(F) toma el limite cuando a=0

Ejemplos

syms x a t h;

limit(sin(x)/x)	1
limit((x-2)/(x^2-4),2)	1/4
limit((1+2*t/x)^(3*x),x,inf)	exp(6*t)
limit(1/x,x,0,'right')	inf
limit(1/x,x,0,'left')	-inf
limit((sin(x+h)-sin(x))/h,h,0)	cos(x)
v = [(1 + a/x)^x, exp(-x)];	
limit(v,x,inf,'left')	[exp(a), 0]

nota: inf es infinito.

Para derivar:

diff(S,'v',n) donde S es una expresión simbólica a derivar, v es la variable con respecto a la cual se quiere derivar y n es el orden de la derivada.

Ejemplo:

diff(sin(x^2)) resulta 2*cos(x^2)*x

diff(t^6,6) resulta 720.

Para Integrar:

INT(S,v) Integra indefinidamente la expresión simbólica S con respecto a la variable v.

INT(S,v,a,b) Calcula la integral definida de la expresión simbólica S con respecto a v entre los limites a y b.

Ejemplo:

syms x x1 alpha u t;

A = [cos(x*t),sin(x*t);-sin(x*t),cos(x*t)];

int(1/(1+x^2)) atan(x)

int(sin(alpha*u),alpha) -cos(alpha*u)/u

int(x1*log(1+x1),0,1) 1/4

int(4*x*t,x,2,sin(t)) 2*sin(t)^2*t-8*t

int([exp(t),exp(alpha*t)]) [exp(t), 1/alpha*exp(alpha*t)]

int(A,t) [sin(x*t)/x, -cos(x*t)/x]

[cos(x*t)/x, sin(x*t)/x] Es la integral de los elementos de la matriz A

TRANSFORMADA Y ANTITRANSFORMADA DE LAPLACE

Matlab calcula con exactitud la transformada y la anti-transformada de Laplace de cualquier expresión simbólica por compleja que sea. En realidad para el uso de este curso la anti-transformada será de mayor utilidad ya que por lo general se trabaja con los circuitos operacionales equivalentes y luego interesa anti-transformar la respuesta para ver su comportamiento en el tiempo.

Para transformar:

L **= laplace**(F) De esta manera guardaremos en la variable L la transformada de Laplace de la expresión simbólica F. Por defecto L va a estar en función de la variable 's' y F tiene que estar en función de 't'.

Ejemplo:

syms a s t w x

laplace(t^5) devuelve 120/s^6

Para anti-transformar:

F=**ilaplace**(L)

Ejemplo:

ilaplace(1/(s-1)) devuelve exp(t)

Residuos en los polos para expansión en fracciones simples.

[r,p,k] = **residue**(b,a) ; entrega en el vector r los residuos en los polos correspondientes al vector p los cuales corresponden a la fracción b/a donde b y a son vectores cuyos elementos corresponden a los coeficientes en orden descendiente de 's' del numerador b y el denominador a. En k se guardan los términos directos.

Si no hay polos múltiples:

$$\frac{B(s)}{A(s)} = \frac{R(1)}{s - P(1)} + \frac{R(2)}{s - P(2)} + \ldots + \frac{R(n)}{s - P(n)} + K(s)$$

Si hay polos multiples:

$$\frac{R(j)}{s - P(j)} + \frac{R(j+1)}{(s - P(j))^2} + \ldots + \frac{R(j+m-1)}{(s - P(j))^m}$$

[b,a] = **residue**(r,p,k) utilizando la funcion de esta manera obtenemos los coeficientes b y a á partir de los residuos en los polos. Es decir que funciona INVERSAMENTE. Ya veremos ejemplos de cómo utilizar esto para síntesis por la forma de Cauer.

GRAFICAS DE FUNCIONES

Vamos a ver ahora como graficar funciones en 2 dimensiones.

Ezplot:

La funcion **ezplot** es una funcion de graficas simplificada. Se usa de las siguientes maneras:

ezplot(f,[a,b]) grafica la funcion f=f(x) entre los limites a < x < b. Si no se especifican a y b Matlab utiliza por defecto a=-2*pi b=2*pi.

ezplot(f, [xmin,xmax,ymin,ymax]) siendo f=f(x,y) una funcion definida implicitamente, grafica f(x,y)=0 entre los limites xmin < x < xmax y ymin < y < ymax. Si esto ultimo no es especificado Matlab usa -2*pi < x < 2*pi y -2*pi <y < 2*pi

ezplot(x,y,[tmin,tmax]) Grafica la curva definida parametricamente por x=x(t) e y=y(t) entre

tmin < t < tmax. Si tmin,tmax no son especificados Matlab usa 0 < t < 2*pi.

Nota: f,x e y pueden ser tanto texto entre ‘ comas’ como expresiones simbólicas utilizando variables ya declaradas.

Ejemplos

ezplot('cos(x)')

ezplot('cos(x)', [0, pi])

ezplot('1/y-log(y)+log(-1+y)+x - 1')

ezplot('x^2 - y^2 - 1')

ezplot('x^2 + y^2 - 1',[-1.25,1.25]); axis equal (esta especificación de axis equal hace que se utilizen los mismos rangos de valores para x y para y , en este caso para que se note que es una circunferencia.

ezplot('x^3 + y^3 - 5*x*y + 1/5',[-3,3])

ezplot('x^3 + 2*x^2 - 3*x + 5 - y^2')

ezplot('sin(t)','cos(t)')

ezplot('sin(3*t)*cos(t)','sin(3*t)*sin(t)',[0,pi])

ezplot('t*cos(t)','t*sin(t)',[0,4*pi])

Nota1: si se hubiesen declarado las variables x,y,t haciendo* syms x y t *no hubiese hecho falta poner las 'comas' en las expresiones.

Nota2: si se quiere graficar una función en el mismo eje coordenado que la anterior se debe escribir HOLD luego de haber graficado la primera función. Para desactivar esta opción escribir nuevamente HOLD.

Nota3: si se quiere conocer el valor de la función en un punto especifico de la grafica, hay que utilizar la función GINPUT, que nos permite señalar en la grafica n puntos y nos devuelve un a matriz nx2 con los pares ordenados correspondientes a los puntos que seleccionamos. Escribir GINPUT, seleccionar los puntos y luego tocar enter.

Nota4: Existe una aplicación para Matlab llamada FUNTOOL , que nos permite graficar funciones de forma muy sencilla y realizar operaciones a modo de calculadora. Tiene ciertas desventajas, pero puede servir en algunos casos. Para ejecutarla escribir FUNTOOL.

Nota5: Si se desea se puede investigar la funcion PLOT(X,Y), la cual grafica los puntos indicados por los pares ordenados correspondientes al vector X versus el vector Y. A veces puede ser de utilidad por que tiene mas opciones en cuanto a el tipo de linea,etc.

FUNCIONES DE TRANSFERENCIA- RESPUESTA TEMPORAL -ANALISIS EN FRECUENCIA

Hay varias formas de definir funciones de transferencia. Hay que aclarar que lo que veremos aquí no es compatible con los métodos simbólicos que vimos anteriormente. Estos métodos se utilizan mucho en Sistemas de Control, para los LTI que son sistemas lineales invariantes en el tiempo.

Definición de función de transferencia a través de coeficientes:

sys=**tf**(num,den) genera una función de transferencia con un numerador y un denominador cuyos términos en potencias de 's' tienen coeficientes correspondientes a los vectores num y den respectivamente.num y den denotan los coeficientes en orden descendiente de las potencias de s.

Ejemplo

tf([1 2 3],[5 8 9])

Transfer function:

s^2 + 2 s + 3

---

5 s^2 + 8 s + 9

Otra forma de usar la función **tf** es asi:

s = **tf**('s');

H = (s+1)/(s^2+3*s+1)

De esta manera disponemos de s como una variable simbólica (aunque no es compatible con los métodos simbólicos vistos anteriormente) y podemos definir funciones de transferencia como la anterior con total facilidad.

Definición de función de transferencia a través de sus ceros, polos y ganancia:

zpk(z,p,k) define una funcion de transferencia cuyos ceros estan dados por el vector z, sus polos por el vector p y su ganancia por el escalar k.

Ejemplo:

zpk([1 9],[-2 0],100)

Zero/pole/gain:

100 (s-1) (s-9)

---

s (s+2)

Nota: si es necesario convertir de **tf** a **zpk** es muy sencillo hacerlo:

Si se quiere pasar de **zpk** a **tf**

F=**zpk**([1 9],[-2 0],100)

Zero/pole/gain:

100 (s-1) (s-9)

---

s (s+2)

D=**tf**(F)

Transfer function:

100 s^2 - 1000 s + 900

---

s^2 + 2 s

y si quiero pasar de **tf** a **zpk**

zpk(D) Zero/pole/gain:

100 (s-9) (s-1)

---

s (s+2)

Polos y Ceros de las funciones de transferencia:

pole(sys) entrega un vector cuyos elementos son los polos de la f.t. sys

zero(sys) entrega un vector cuyos elementos son los ceros de la f.t. sys

pzmap(sys) entrega un mapa de los ceros y los polos de la función en el plano complejo.

Respuesta temporal

step(sys) : Grafica la respuesta al escalon unitario de el sistema sys, el que debe crearse ya sea con la funcion **tf** o con **zpk**.

impulse(sys) : Grafica la respuesta al impulso de el sistema sys, el que debe crearse ya sea con la funcion **tf** o con **zpk**.

lsim(sys,u,t) : Grafica la respuesta del sistema sys a la entrada u valuando en el vector equi-espaciado t. (nos evita antiransformar)

Ejemplo:

t = 0:0.01:5; u = sin(t); lsim(sys,u,t)

Esto va a graficar la salida del sistema sys cuando en su entrada tiene una función senoidal durante 5 segundos.

Respuesta en frecuencia

bode(sys) : Grafica el diagrama de Bode del sistema sys.

nyquist(sys) : Grafica el diagrama de Nyquist del sistema sys. (tener en cuenta que si hay polos en el origen la funcion NO CIERRA el diagrama

Nota: Una vez que las respuestas están graficadas, pulsando el boton derecho del mouse sobre la grafica aparecen opciones.

Zoom: para acercarse y/o alejar la vista.

Peak Response: Marca el pico de la grafica.

Grid: Genera una grilla sobre la grafica.

Closedloop response: Da el diagrama de Bode o de Nyquist del sistema a lazo cerrado . (sys/(1+sys))

Recordemos la funcion **ginput** que nos permitira saber el valor de la respuesta en cualquier punto.

Vale la pena aclarar que hay muchas mas opciones, y hay formas mas sencillas de hacer lo mismo como **rltool** (para el lugar de las raices) y muchas otras, pero requieren conceptos de Control, por lo tanto no se veran en este curso.

FILTROS MODERNOS –CHEBYSCHEV Y BUTTERWORTH

Butterworth

[b,a] = **butter**(n,wn,'high', 's') genera los coeficientes del numerador **b** y el denominador **a** de un filtro de Butterworth de orden **n** con frecuencia de corte **wn**. La denominación **'high'** inidca que es un filtro pasaaltos, si se escribe **'low'** se crea un pasabajos, si se escribe **'stop'** un eliminabanda y si se escribe **'bandpass'** un pasabanda. En los dos ultimos casos notese que wn=[w1 w2], es decir que es un vector de dos elementos que indican en ancho de banda de paso o eliminación. La **'s'**, solo indica que se desea hacer un filtro analógico.

[z,p,k] = **butter**(...) si se pide un resultado de tres variables, se entregaran los ceros **z** los polos **p** y la ganancia **k** del filtro que se pida.

n = **buttord**(Wp, Ws, Rp, Rs,'s') entrega el minimo orden **n** del filtro de Butterworth necesario para tener una perdida máxima de Rp dB en la banda de paso y una atenuación de Rs dB en la zona de atenuación. Wp y Ws denotan las frecuencias de transmisión y atenuación respectivamente. Si se quiere diseñar un filtro pasabajos se puede elegir Wp=0.01 y Ws=frecuencia de corte deseada, lo mismo si se desea un pasaaltos Wp=frecuencia de corte deseada y Ws=0.01.

Chebyschev:

[b,a] = cheby1(n,r,wn,'high','s') genera los coeficientes del numerador **b** y el denominador **a** de un filtro de Chebyschev de orden **n** con frecuencia de corte **wn** y riple de **r dB**. La denominación **'high'** inidca que es un filtro pasaaltos, si se escribe **'low'** se crea un pasabajos, si se escribe **'stop'** un eliminabanda y si se escribe **'bandpass'** un pasabanda. En los dos ultimos casos notese que wn=[w1 w2], es decir que es un vector de dos elementos que indican en ancho de banda de paso o eliminación. La **'s'**, solo indica que se desea hacer un filtro analógico.

[z,p,k] = **cheby1**(...) si se pide un resultado de tres variables, se entregaran los ceros **z** los polos **p** y la ganancia **k** del filtro que se pida.

n = **cheb1ord**(Wp, Ws, Rp, Rs,'s') entrega el minimo orden **n** del filtro de Butterworth necesario para tener una perdida máxima de Rp dB en la banda de paso y una atenuación de Rs dB en la zona de atenuación. Wp y Ws denotan las frecuencias de transmisión y atenuación respectivamente. Si se quiere diseñar un filtro pasabajos se puede elegir Wp=0.01 y Ws=frecuencia de corte deseada, lo mismo si se desea un pasaaltos Wp=frecuencia de corte deseada y Ws=0.01.

CIRCUITMAKER 2000

INTRODUCCIÓN

CircuitMaker es un programa de simulación Digital y Analógica que es cada vez mas usado en las universidades de todo el mundo por estudiantes. Esto se debe a que es muy facil de usar, es intuitivo y tiene cubiertas casi todas las necesidades de simulación que un estudiante puede tener a lo largo de su carrera. En este curso veremos solo la simulación analógica, que es la que nos hace falta para poder simular redes pasivas. Pero al saber hacer simulaciones analógicas el alumno quedará a un paso de poder hacer simulaciones digitales ya que estas son mucho mas sencillas.

CircuitMaker es un programa tan sencillo que la mejor forma de aprender a usarlo es usándolo, con esto quiero decir que no es conveniente dar muchas explicaciones escritas, sino comenzar a usarlo para casos concretos.

DIBUJO DE CIRCUITOS

Para usar circuitmaker hay que tener en cuenta que, sobre el esquema, se utilizan 6 tipos de cursores distintos.

Arrow: Es el icono de la flechita. Es el que nos permitirá una vez dibujado el circuito elegir opciones, cambiar valores, etc.

Wire Tool: Es el icono de la cruz. Este cursor nos permitirá dibujar los cables que vamos a utilizar. Podemos dibujar los cables de todo el circuito sin poner componentes y luego poner los componentes encima de los cables, Circuitmaker hará las conexiones por nosotros.

Text Tool: Es el icono de la letra A. Este cursor nos permitirá escribir texto sobre el esquema.

Delete Tool: Es el icono del rayito. Este cursor es para borrar elementos del circuito. Cables, componentes,etc.

Probe Tool: Es el icono con forma de punta lógica. Sirve para realizar todas las mediciones.

Zoom Tool: Es el icono de la lupa. Sirve para acercarse. Si se presiona shift sirve para alejarse.

En la parte izquierda de la pantalla se observa una ventana que tiene en su parte superior dos solapas:

Browse: Es un inventario de todos los componentes que podemos utilizar en Circuitmaker. Es una base de datos enorme donde hay desde resistencias hasta circuitos integrados complejos. Nosotros utilizaremos la parte

llamada **General** . Ahí esta la opcion de seleccionar resistores (resistors) inductores (inductors) capacitores.(capacitors) ,fuentes de DC y masas o tierras (sources). Se pueden seleccionar de esta ventana y arrastrarlos hacia el esquema. Otra forma, que es la recomendada es utilizar las abreviaturas de teclado:

r – nos entrega un resistor

c – nos entrega un capacitor

l- nos entrega un inductor

b – nos entrega una batería

0- entrega una masa o una tierra.

g- nos entrega el generador de señales

Para cambiar el valor de los componentes hay que seleccionar el cursor Arrow Tool y hacer doble clic sobre el componente. Luego donde dice Label-Value modificar el valor al que uno desee.

- **Practicar ahora el dibujo de circuitos RLC alimentados con el generador de señales y referenciados a tierra. Para verificar si estan bien conectados utilizar la opcion Simulation-Check wire connections (marcará en rojo los cables desconectados). Si se equivoca utilice el cursor Delete, borre lo que esta mal y vuelva a conectar.**

- **Recordar siempre colocar la referencia a tierra!!!**

- **Si se desea rotar un componente (MUY UTIL) una vez seleccionado presionar el boton derecho del mouse.**

GENERADOR DE SEÑALES

Haciendo doble clic sobre el generador de señales obtendremos una serie de opciones que nos permitirán configurar la señal que queremos utilizar en nuestra simulación.

DcOffSet: Es la componente de continua sobre la cual montaremos nuestra señal.(no lo utilizaremos en este curso)

Peak Amplitude: Es el pico de amplitud que queremos que la señal tenga.

Frequency: La frecuencia de la señal.

Start Delay: Demora en el comienzo. Comienza luego de xxx segundos.

Damping Factor: Factor de amortiguamiento. ξ

Output: Definir si queremos generador de corriente o de voltaje.

Wave: Nos permite elegir el tipo de señal.

SIMULACIÓN

Para comenzar la simulación de un circuito en Circuitmaker se debe pulsar el icono que tiene el dibujo de una onda senoidal. Una vez pulsado utilizando el cursor Probe Tool se pueden realizar mediciones de distinto tipo. Pulsando con el Probe Tool sobre un nodo (V) se medirá **voltaje**. Pulsando sobre el borde de algun elemento de circuito el Probe Tool mostrará (I), por lo tanto se estara midiendo la corriente que pasa por ese elemento, y si se pulsa sobre un elemento de circuito y la probe tool marca (P) se estará midiendo la potencia disipada por ese elemento.

Transient Analysis (Osciloscopio)

Si el circuito que estamos midiendo está alimentado por el generador de señales las mediciones antes mencionadas serán mostradas en forma de un grafico de respuesta temporal.

Operating Point (no se usara en el curso)

OPCIONES DE SIMULACIÓN

Para ver mas opciones de simulación pulsar sobre el icono que tiene la llave. De este menu aprenderemos a usar las siguientes:

DC : Nos permitirá realizar graficas de un parámetro que varia con respecto a uno que medimos. Por ejemplo si armamos un circuito con jfet y seleccionamos que varíen los voltajes, podremos medir las corrientes, y obtener automáticamente las curvas características. (no es de utilidad para teoría de redes)

AC: Nos permitirá realizar graficas de respuesta en frecuencia de el parámetro medido. Utilizando esta opcion combinada con otras que veremos en ejemplos posteriores, podremos obtener el diagrama de BODE de un circuito a través de su medición.

Transient/Fourier : Nos permite configurar los tiempos de análisis que se desean tomar en al **Transient Analysis.** Por lo general esta opción no necesita ser configurada. Pero veremos ejemplos en los cuales, necesitaremos hacer algunas modificaciones para lograr un resultado mas satisfactorio.

Transfer Function : Activar esta opcion nos dara la posibilidad de medir impedancias de Exitación y de Salida, y obtener la funcion de transferencia (numérica) entre la entrada seleccionada y el parámetro medido.

WAVE – OPCIONES SOBRE LAS CURVAS DE RESPUESTA

Cuando estamos simulando, a nuestra izquierda veremos una serie de opciones que podremos aplicar sobre las graficas de respuesta.

- **Si se desean realizar varias mediciones simultaneas presionar la tecla Shift mientras se hacen las mediciones.**

Las opciones son:

View:

Single Cell: Ver las mediciones una por una. (si se realizan varias mediciones sus curvas se superpondrán)

All Cells: Ver todas las mediciones juntas.

Scaling:

Nos permite seleccionar la escala en X, en Y . Fit X hace que la curva se vea toda a lo largo del eje X, Fit Y lo mismo con respecto al eje Y. Auto Y (recomendada) hace que el eje Y se maneje automáticamente.

Measurement Cursors:

Sirve para realizar mediciones sobre los gráficos. Se elije a que nodo pertenecerá el nodo 1 y 2. Luego se posicionan los nodos sobre la grafica dependiendo del tipo de medicion. Se pueden restar las valores (cursor2 – cursor1), se puede sacar el promedio entre ambos (Average) El mínimo y el máximo, la RMS (Raíz media cuadrática), etc.

Opciones Botón derecho:

Presionando el boton derecho del mouse sobre las graficas, aparecerán opciones de Zoom (acercar y alejar), Preferences que es para elegir colores de curvas, Scalling que nos permite elegir las escalas en X e Y (luego veremos como usar esto para obtener diagramas de Bode), Math que nos permite realizar operaciones matemáticas sobre las respuestas.

OPCIONES GENERALES

Imprimir: Si se desea imprimir el circuito File-Print Schematic. Si se desea imprimir las curvas File-Print Waveforms.

Guardar: File-Save As.

Abrir: File – Open

CONSIDERACIONES

No vale la pena seguir viendo opciones del programa, sin comenzar a utilizarlas. En la sección que sigue realizaremos Ejercicios de la guía de Trabajos Prácticos de Teoría de Redes, y aprenderemos a utilizar mejor este programa y a comparar los resultados obtenidos por Matlab, para corroborar nuestros Ejercicios hechos a mano.

EJERCICIOS DE LA GUÍA DE TRABAJOS PRÁCTICOS

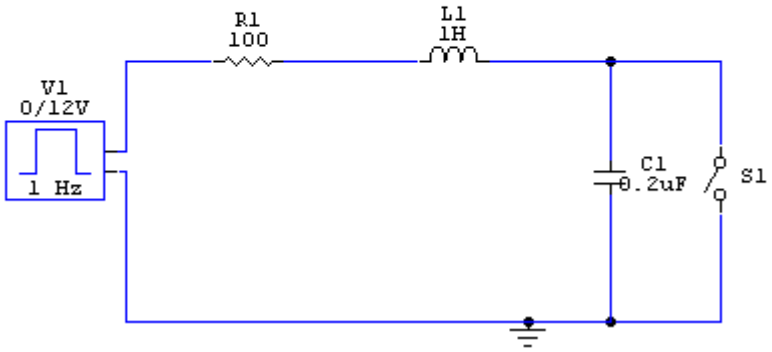
Pasos a seguir:

- Resolver el ejercicio manualmente, sin utilizar la computadora para nada.
- Simular el circuito, interpretar los resultados, y comparar.
- Comprobar los calculos con Matlab

RESPUESTA TEMPORAL

Ejercicio 1.13

Calcular y graficar i(t); $v_c(t)$ y $v_L(t)$. En t=0 abre S1



El circuito que pide el ejercicio está alimentado por una batería, aquí para poder simularlo utilizaremos el generador de señales para poder obtener una respuesta temporal en la simulación.

Dibujo del esquema:

Para dibujar el circuito haremos como ya se ha visto. La llave hay que buscarla utilizando SEARCH a la izquierda de la pantalla. Su nombre es **spst switch**.

Simulación:

Cuando utilizamos una batería en teoría de redes, en realidad matemáticamente, estamos calculando la respuesta al escalón del sistema. Por lo tanto para simularlo es conveniente crear con el generador de señales un pulso con un periodo grande y un ancho largo para ver la respuesta del sistema.

Esto se hace con doble clic sobre el generador de señales, luego seleccionamos **wave**. Hecho esto pulsamos sobre **pulse.** Ahora aparecerán opciones para configurar el pulso. **Pulse Amplitude**, es la amplitud del pulso, la cual debe ser el valor de la bateria,**Period** es el periodo para el cual recomiendo poner 1 segundo. **Pulse Width** es el ancho del pulso, el cual es conveniente fijarlo en un valor cercano al segundo.

Luego debemos ir a la configuración de simulación (Analyses setup) (icono llave), y en la opcion transient fourier fijar los siguientes valores:

Start Time: (tiempo inicio) 0 S

Stop Time: (tiempo final) 50.00mS

Step Time y Max Step : 1.000mS

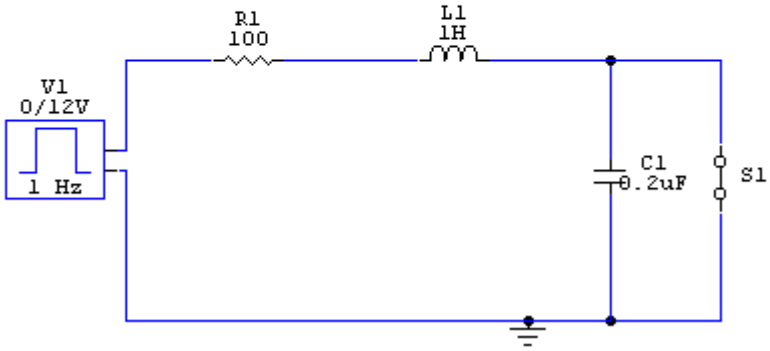
De esta manera, a los fines del estudio que realizaremos, el generador de señales estará funcionando como una batería en nuestro tiempo de análisis.

Comenzaremos a resolver el ejercicio:

En t=0 se abre la llave, por lo tanto en t<0 en circuito será un RL. Hay que tener en cuenta que Circuitmaker no guarda las condiciones finales de los elementos, es decir que si un capacitor se carga, cuando se vuelve a simular

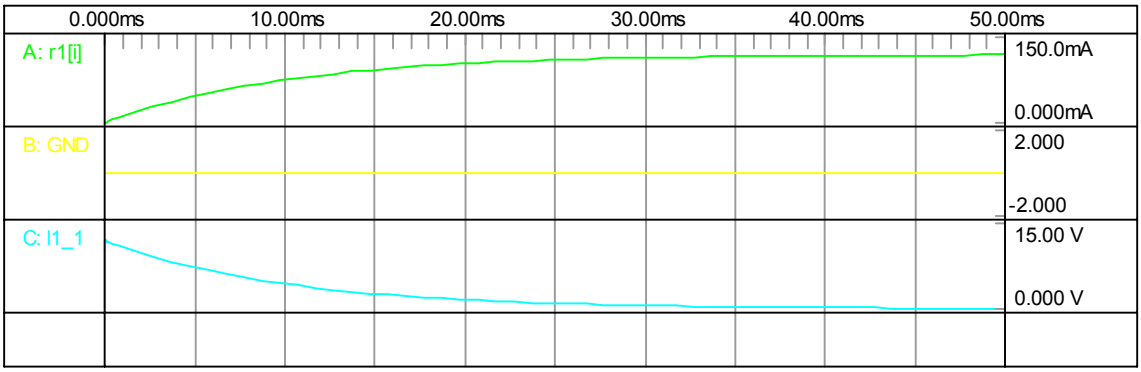
comienza nuevamente con el capacitor descargado. Por esto veremos mas adelante como utilizar un elemento llamado .IC que nos permitirá asignar condiciones iniciales a un nodo.

En t<0



Se nos pide i(t); $v_c(t)$ y $v_L(t)$, por lo tanto manteniendo apretado shift pulsaremos sobre el borde de algun elemento con el cursor (Probe Tool) de forma tal que este indique la letra (I), como es un circuito serie esta será la i(t) del circuito. Para medir $v_c(t)$ y $v_L(t)$ pulsaremos en los nodos correspondientes y veremos que la Probe Tool marca (V). Es conveniente graficarlos en celdas distintas, por lo tanto elegir la opción **All Cells.**

Al hacer esto veremos los siguientes graficos:



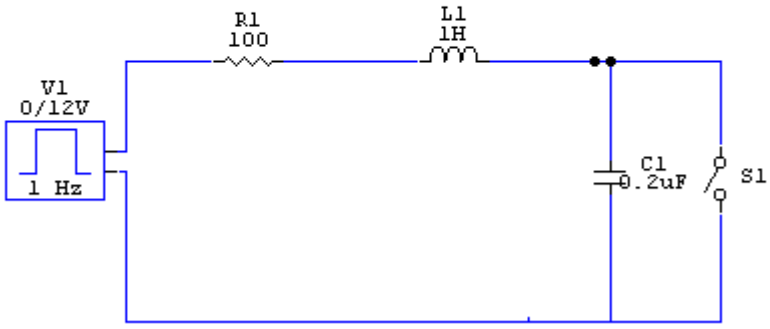
El grafico en verde corresponde a i(t) , el grafico en amarillo a $v_c(t)$ que obviamente es 0 y el celeste a $v_L(t)$. Todos para t<0.

Para continuar el analisis debemos aplicar un poco de nuestro conocimiento ya que, como vimos anteriormente, Circuitmaker no guarda las condiciones finales del circuito.

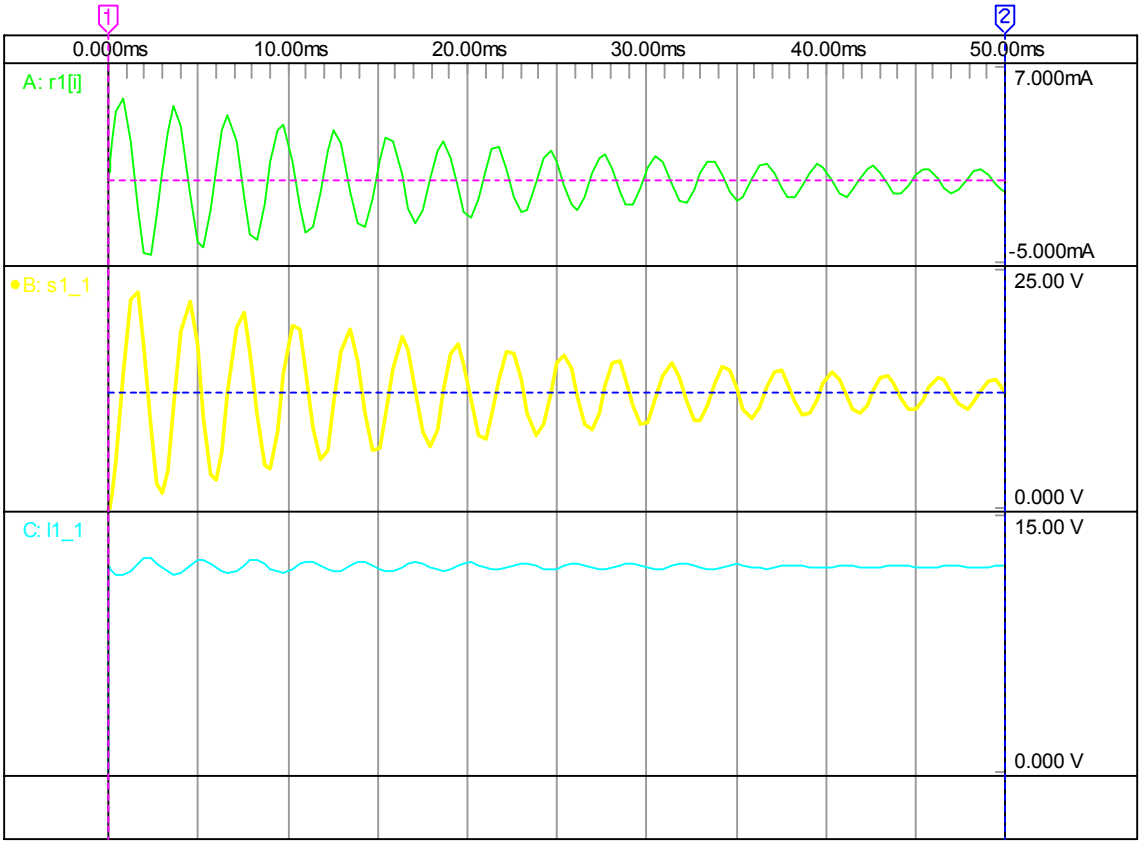
Vemos que la corriente final tiende a i_0=120.0mA, y sabemos que si el circuito cambia esta corriente tendra influencia en el nuevo circuito la cual podra considerarse como una fuente de valor $L*i_0$. Este tipo de condiciones iniciales (las del inductor) no se pueden plantear con Circuitmaker de manera sencilla, por lo que es conveniente tener en cuenta que existen y saber interpretar los resultados como sigue:

Cancelamos la simulación, cerramos la llave (presionando sobre ella) y volvemos a simular. Circuitmaker no tendrá en cuenta las condiciones iniciales, pero a esto lo supliremos con una interpretación correcta de la respuesta que obtenemos.

En t>0 tenemos:



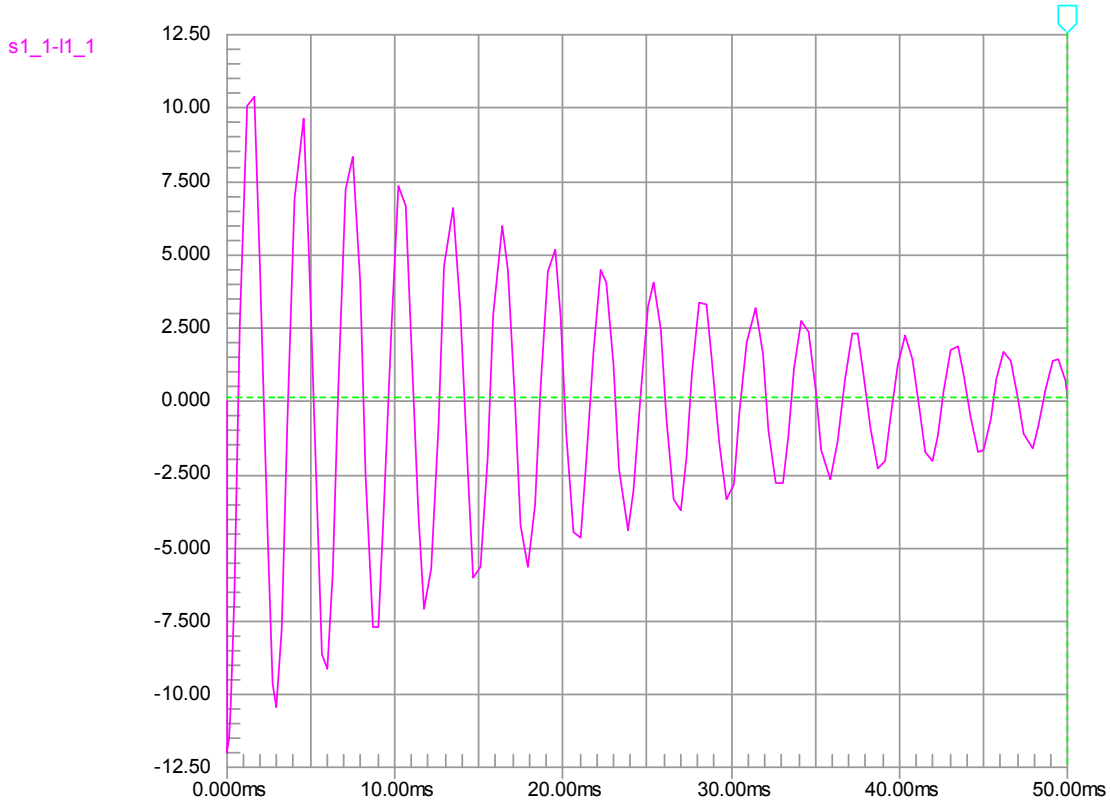
Simulando obtendremos:



Measurement Cursors

1	r1[i]	X: 0.0000	Y: 0.0000
2	s1_1	X: 50.000m	Y: 12.180
Cursor 2 - Cursor 1		X: 50.000m	Y: 12.180

El grafico en verde corresonde a i(t), el amarillo a $v_c(t)$ y el celeste corresponde a un voltaje medido desde el inductor con respecto a la masa. Para obtener v_L tenemos que utilizar la opcion math y hacer la resta $v_c(t)$ (curva amarilla) menos la curva celeste. Entonces v_L quedará:



Measurement Cursors

1	r1[i]	X: 0.0000	Y: 0.0000
2	s1_1-I1_1	X: 50.000m	Y: 113.58m
Cursor 2 - Cursor 1		X: 50.000m	Y: 113.58m

Como veremos mas adelante con Matlab, y como ya intuíamos por la falta de las condiciones iniciales esta no es la verdadera respuesta del sistema que nos interesa, pero pensemos como es:

- Notamos que la corriente debería empezar desde el valor i_0, en vez de empezar desde 0.
- $v_c(t)$ va a ser la suma de la repuesta propia (sin C.I.) mas otra curva senoidal decreciente que tiende a cero en régimen debida a la condición inicial (esto se puede ver en las ecuaciones, como ya comprobaremos con Matlab). Por lo tanto la curva real deberá ser parecida, pero tendrá picos negativos.
- $v_L(t)$ va ser 0 en t=0 , y los picos seran de mayor amplitud.
- **Tener en cuenta que la Probe Tool mide siempre con respecto a la masa.**

Para comparar los resultados obtenidos anteriormente utilizaremos Matlab:

En la Command window podemos escribir

s=**tf**('s');

v=12

```
v =

    12

» c=0.2e-6

c =

      2e-007

» r=100

r =

   100

» io=120e-3

io =

        0.12

» l=1

l =

     1

» I=(v/l + io*s)/(s^2+r*s/l+1/(l*c))
```

Transfer function:

```
    0.12 s + 12

--------------------

s^2 + 100 s + 5e006
```

```
» Vc=I/(s*c)
```

Transfer function:

```
        0.12 s + 12

----------------------------

2e-007 s^3 + 2e-005 s^2 + s

Vl=I*s*l
```

Transfer function:

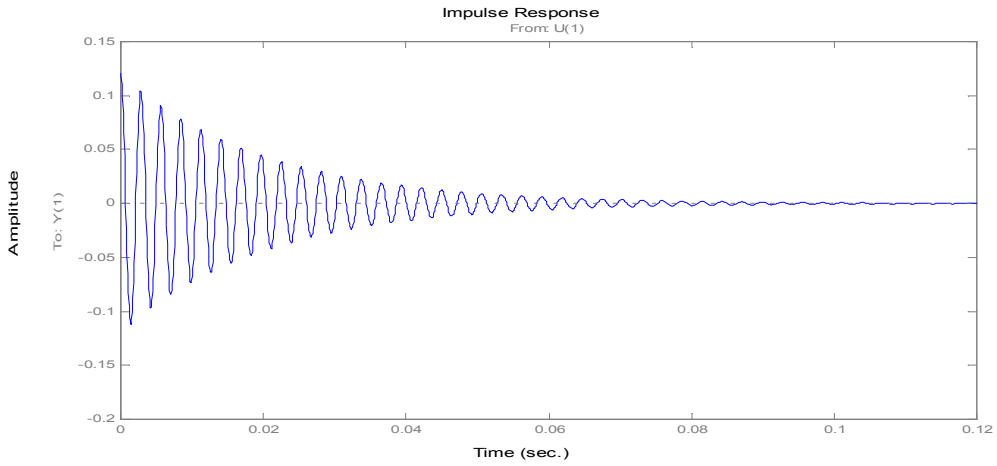
0.12 s^2 + 12 s

---

s^2 + 100 s + 5e006

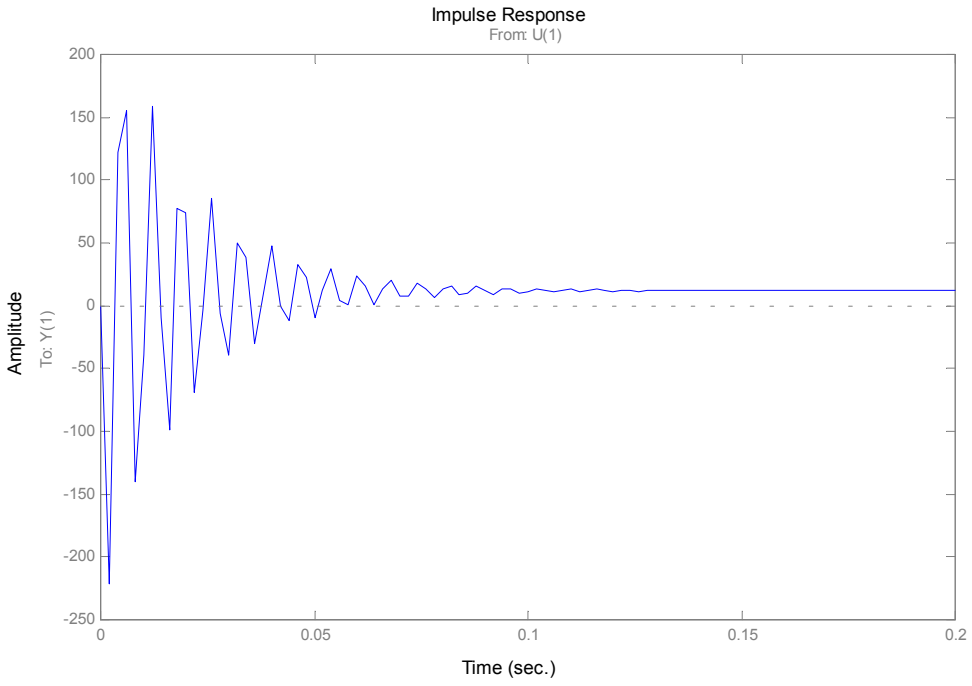
Haciendo esto ya tenemos I(s) $V_C(s)$ y $V_L(s)$, y para ahorrarnos la antitransformada de esto es conveniente utilizar la función impulse, y obtener la curva de respuesta al impulso del sistema la cual seria equivalente a la curva de la funcion antitransformada:

Impulse(I) nos entrega:



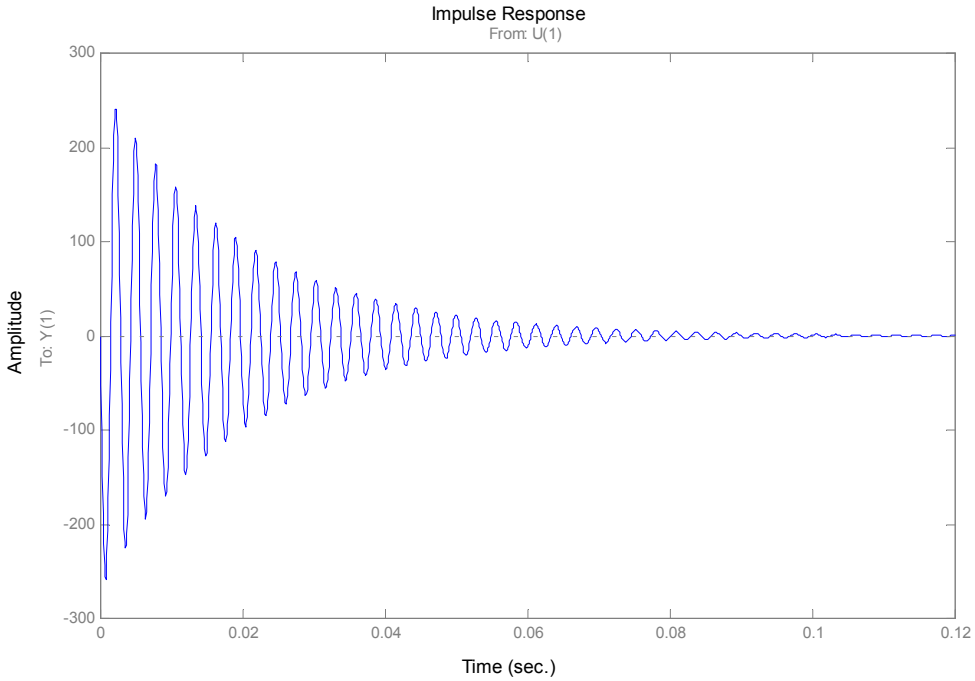
Notamos, como esperábamos, que esta curva empieza desde 0.12 (podrían comprobarlo con ginput) que es el valor $i_{0.}$

Impulse(Vc) entrega:



Podemos ver que aparecen los picos negativos, pero la tendencia en régimen es de 12 V, como esperábamos.

Impulse(Vl) entrega:



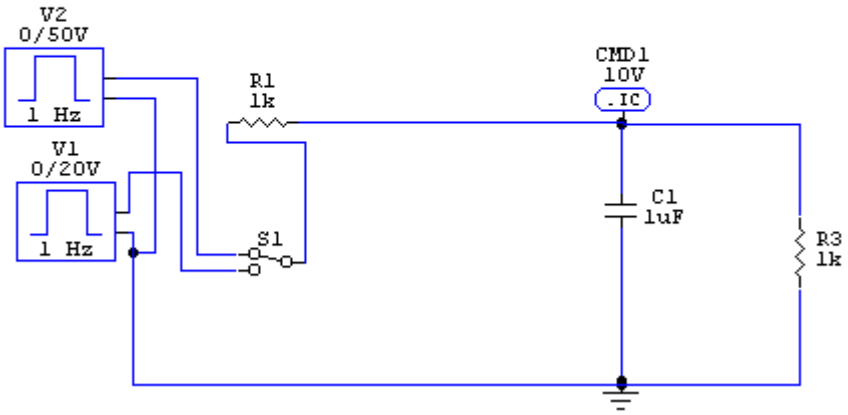
Notamos que las condiciones iniciales influyen sobre la amplitud de los picos de la respuesta, aumentándolos y tambien que en t=0 V_L=0, cuando si no hubiese habido C.I el inductor se comportaria en t=0 como un circuito abierto como muestra la respuesta obtenida con Circuitmaker.

El que haya resuelto este problema a mano notara el alivio que significa poder reolverlo asi, ya que cuando queremos calular v_L y utilizamos el teorema del valor final, notamos que el limite dá infinito lo cual no es lógico. Lo que pasa es que para ese caso el **teorema del valor final no puede usarse,** ya que el limite de $V_L(t)$ cuando t--->inf no existe.

Ejercicio 1.14

Este ejercicio es mucho mas sencillo que el anterior, y con los resultados de la simulación ya nos veremos satisfechos. Ademas nos servira para aprender a usar otra funcion de Circuitmaker que es la **.IC** que es un elemento que podemos conectar al circuito para simular con condiciones iniciales del voltaje en un capacitor.

Al circuito lo dibujamos asi:



Esa llave doble se llama **spdt switch** .Los generadores de señales se configuran en forma similar al ejercicio anterior.

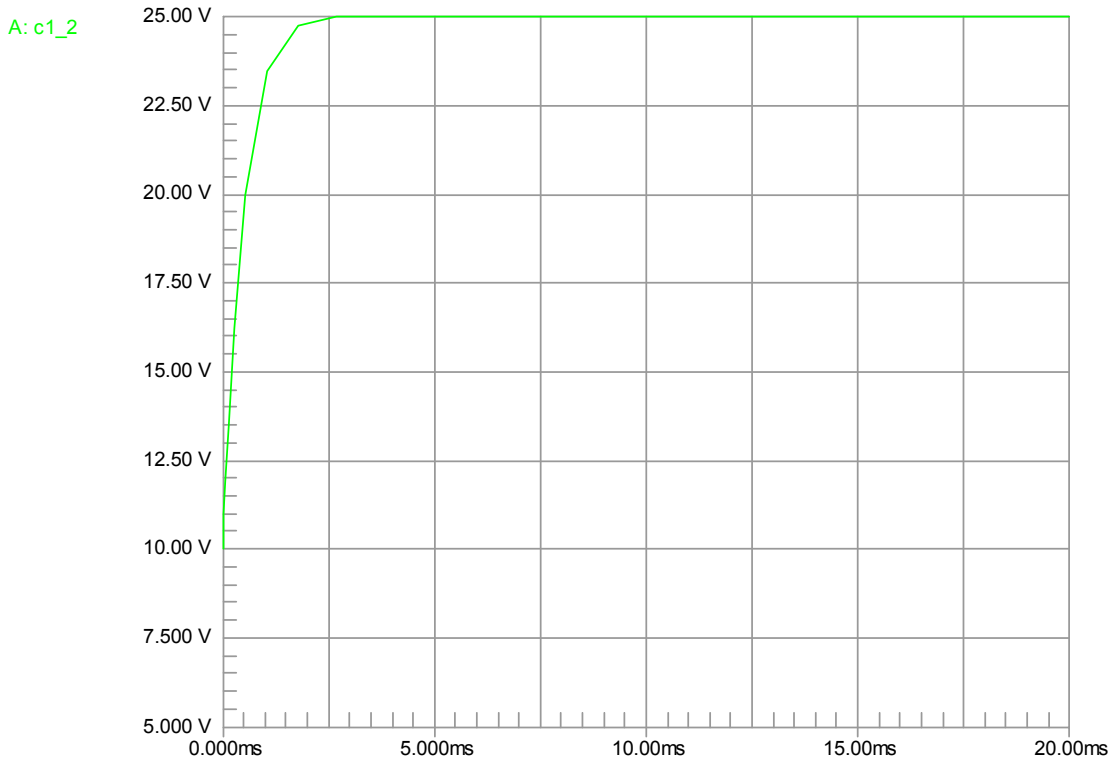
t>0

Si simulamos el circuito conectado a V1 será un circuito RC con una resistencia en paralelo y veremos que en regimen tendra un voltaje v_0=10 V

t>0

Ahora el capacitor tendra a sus bornes como voltaje inicial v_0=10V. Esto se puede simular utilizando el elemento .IC, que se puede buscar en search.

Conectando este elemento en el nudo correspondiente al capacitor (ver figura) se simulara el circuito con las condiciones iniciales y obtendremos



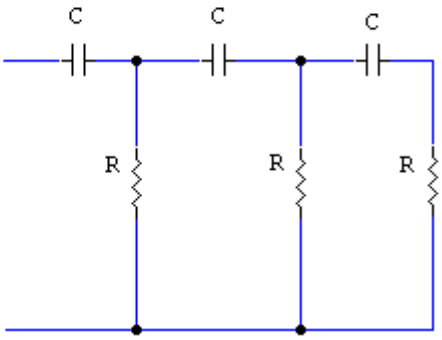
Este resultado esta perfecto, y corresponde a la realidad, ya que lo hemos simulado teniendo en cuenta sus condiciones iniciales.

FUNCIONES DE TRANSFERENCIA Y RESPUESTA EN FRECUENCIA

Matlab es una herramienta muy poderosa para trabajar con variable simbolica. Por lo tanto calcular funciones de transferencia con Matlab es muy sencillo.

Ejercicio 4.4

Hallar la funcion de transferencia V_2/V_1, utilizando alguno de los metodos conocidos para ello.



Este tipo de ejercicios, si no son encarados correctamente desde el principio, suelen ser muy engorrosos de hacer manualmente. De todas maneras es interesante intentar hacerlos, y luego comprobar con Matlab si nuestro sacrificio fue en vano o no.

Hay que armar la matriz de impedancia del método de las mallas, deducir algunos aspectos teóricos, y el ejercicio está resuelto.

Creamos la matriz de impedancia de mallas:

» **syms** s

» **syms** r c real

» A=[(r+(1/(s*c))) -r 0;-r ((1/(s*c))+2*r) -r;0 -r ((1/(s*c))+2*r)]

A =

[r+1/s/c, -r, 0]

[-r, 1/s/c+2*r, -r]

[0, -r, 1/s/c+2*r]

Para ver mejor la matriz A podemos usar la funcion **pretty(A)** y vermos el resultado mas legible.

Si llamamos D al determinante de la Matriz A.

Notamos que $I_3=V_1R^2/D$

Y teniendo en cuenta que $V_2=I_3R$

Vemos que $V_2/V_1=R^3/D=G$

Calculemos D

» D=**det**(A)

D =

(5*r*s*c+6*r^2*s^2*c^2+r^3*s^3*c^3+1)/s^3/c^3

entonces

» G=r^3/D

G =

r^3/(5*r*s*c+6*r^2*s^2*c^2+r^3*s^3*c^3+1)*s^3*c^3

Para ver bien el resultado escribir **pretty(G)**

Si se quisiera obtener el diagrama de Bode de esa funcion de transferencia hacer lo siguiente

Hay que limpiar el workspace con clear, luego asignarle valores a R y a C, copiar con el mouse la expresión obtenida anteriormente y crearla ahora como funcion de transferencia.

»clear

» r=100

r =

100

c=1e-6

c =

1.0000e-006

s=**tf**('s')

Transfer function:

s

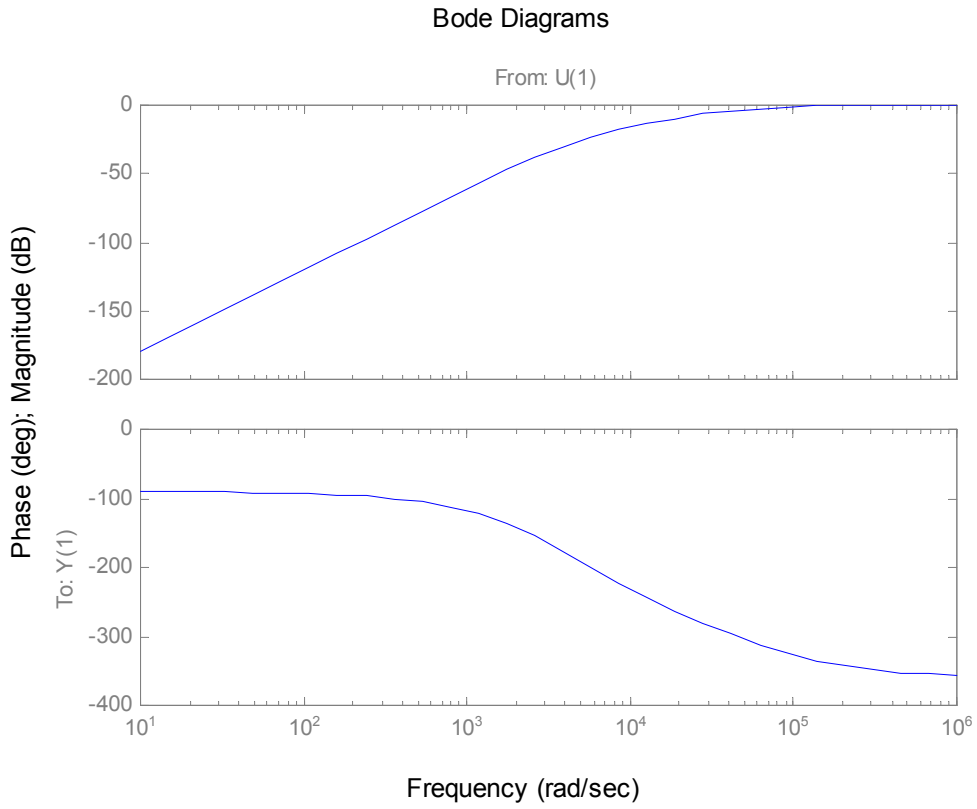
G=r^3/(5*r*s*c+6*r^2*s^2*c^2+r^3*s^3*c^3+1)*s^3*c^3

Transfer function:

1e-012 s^3

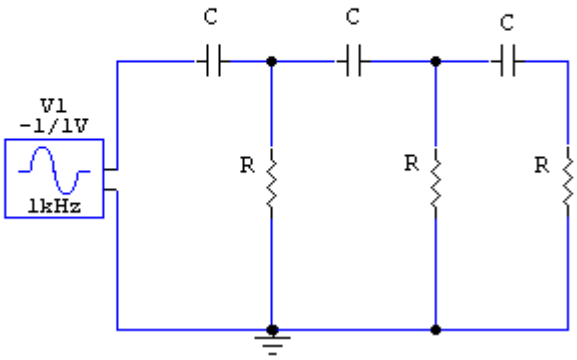
1e-012 s^3 + 6e-008 s^2 + 0.0005 s + 1

ahora escribimos **bode(G)** y obtenemos el diagrama de bode correspondiente.



Si queremos obtener el diagrama de Bode de este mismo circuito a través de su simulación en Circuitmaker debemos hacer lo siguiente:

Dibujar el circuito:



asignar los valores R=100 y C=1uF

Luego en Analyses Setup (icono llave) configurar AC de la siguiente manera:

Start Frequency: 1.000 Hz

Stop Frequency: 10000Hz

Estas son las frecuencias en Hz (OjO con eso) que definen el rango de análisis.

Test Points: siempre poner un valor superior a 20

En Sep elegir **Decade**

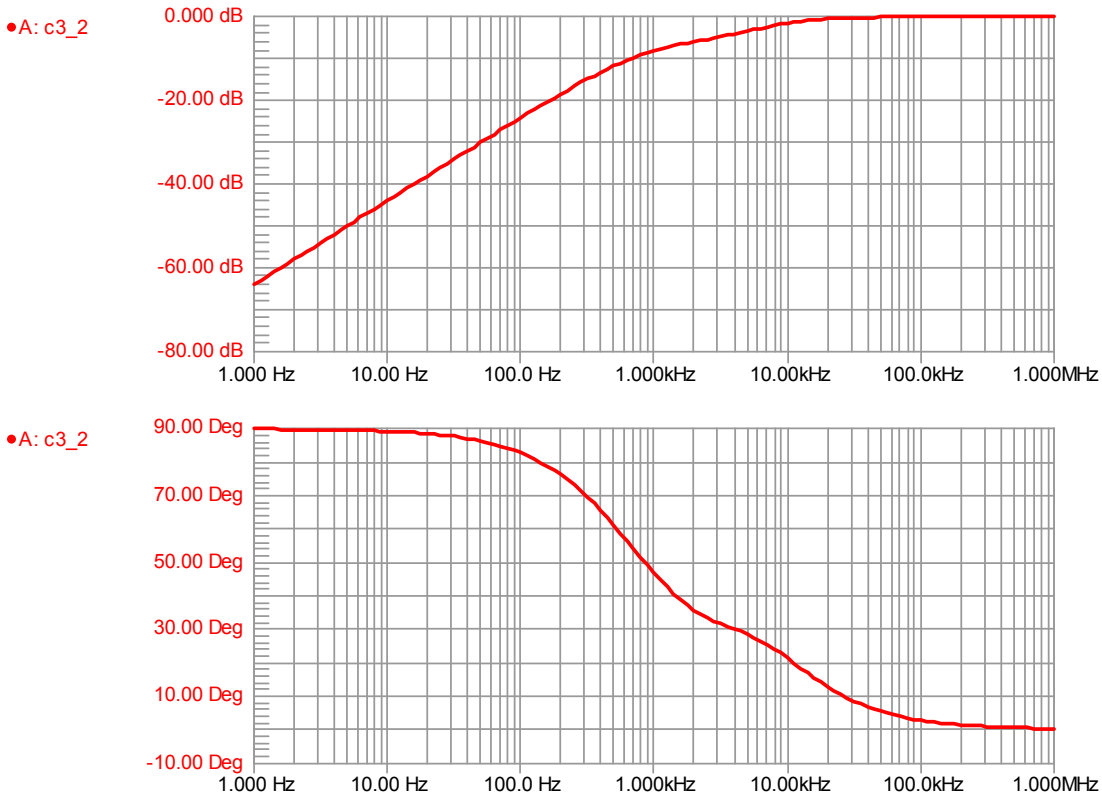
Comenzar el analisis. Si se quiere V_2/V_1 pulsar con la **Probe Tool** sobre el nodo correspondiente a V_2 , y como el análisis es con respecto a la alimentación V_1, obtendremos el grafico de ganancia de Voltaje que buscamos.

Una vez hecho esto pulsar con el boton derecho del mouse sobre el borde del grafico y elegir **scaling** aquí elegir en Y axis:

Primary: Magnitude in Decibels

Secondary: Phase in Degrees

Esto convierte el grafico de Ganancia en un Diagrama de Bode de Modulo Fase



- **Notamos la diferencia en los gráficos de Fase de Matlab y el de la Circuitmaker. A pesar de que son diferentes ambos están bien, ya que su cambio de fase es equivalente (0º y –360º tienen el mismo significado físico)**

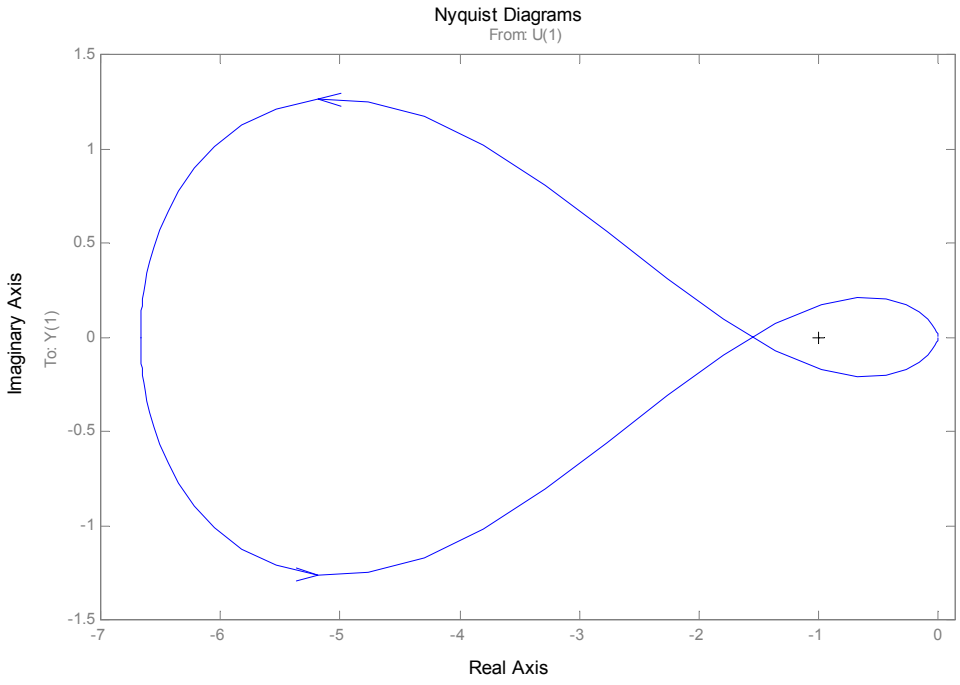
Diagramas de Nyquist

Matlab realiza diagramas de Nyquist perfectamente mientras las funciones no tengan polos en el origen, dado este caso Matlab no cierra los diagramas, y esto habrá que hacerlo manualmente.

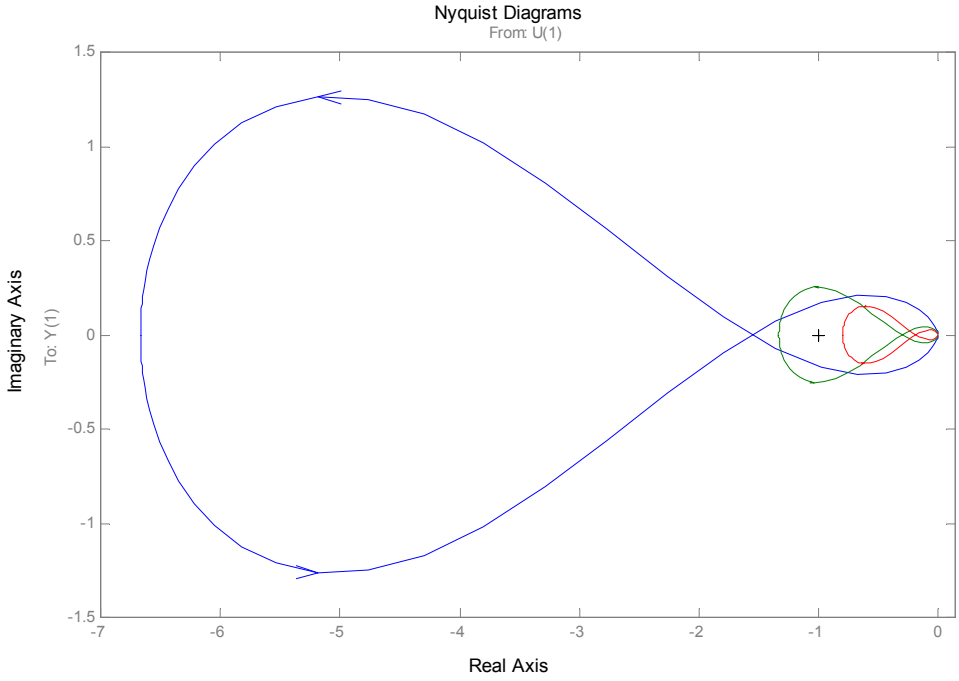
Ejercicio 5.25

Defino la funcion de transferencia asi:

```
k=100

k =

    100


» GH=zpk([],[1 -3 -5],k)

Zero/pole/gain:

        1

 ----------------

(s-1) (s+3) (s+5)

 » nyquist(GH)  %nos entrega:
```



Ahora utilizando la funcion **HOLD** podemos graficar en los mismos ejes distintos diagramas para distintos K, y observar para cual es estable:



- Observamos que es estable para 15<K<64 , y notamos que de los 3 diagramas el unico estable es el que está en verde que tiene K en ese rango.

SÍNTESIS

Para realizar los ejercicios de síntesis por la 1ª forma de Foster, poseemos una herramienta en Matlab que nos calcula los residuos en los polos de la función que queramos sintetizar.

Ejercicio 8.1

syms s

» F=((s^2+1)*(s^2+9))/(s*(s^2+4))

F =

(s^2+1)*(s^2+9)/s/(s^2+4)

»F=simple(F);

» F

F =

(s^4+10*s^2+9)/(s^3+4*s)

[num,den]=**numden**(F)

num =

s^4+10*s^2+9

den =

```
s*(s^2+4)

» num=sym2poly(num)

num =

    1     0    10     0     9

» den=sym2poly(den)

den =

    1     0     4     0

» [r,p,k]=residue(num,den)


r =

   15/8

   15/8

    9/4



p =

     0      +    2i

     0      -    2i

     0



k =

     1           0
```

Tener en cuenta que los vectores r, p y k son los residuos en los polos y los términos directos respectivamente.

FILTROS

Como ya hemos visto, podemos crear filtros modernos con Matlab. Ahora veremos como simular filtros con Circuitmaker y obtener su grafica de Atenuación.

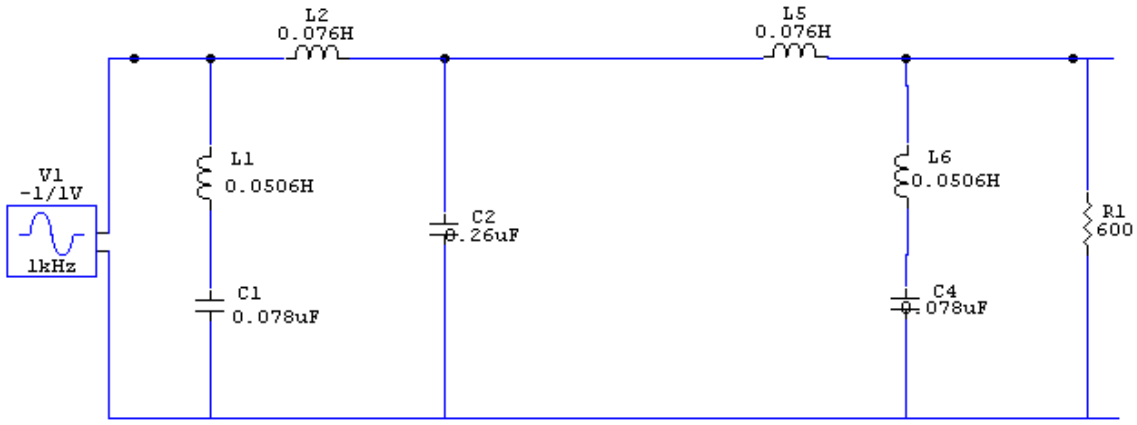
Ejercicio 9.11

Nos pide realizar un filtro compuesto pasabajos con:

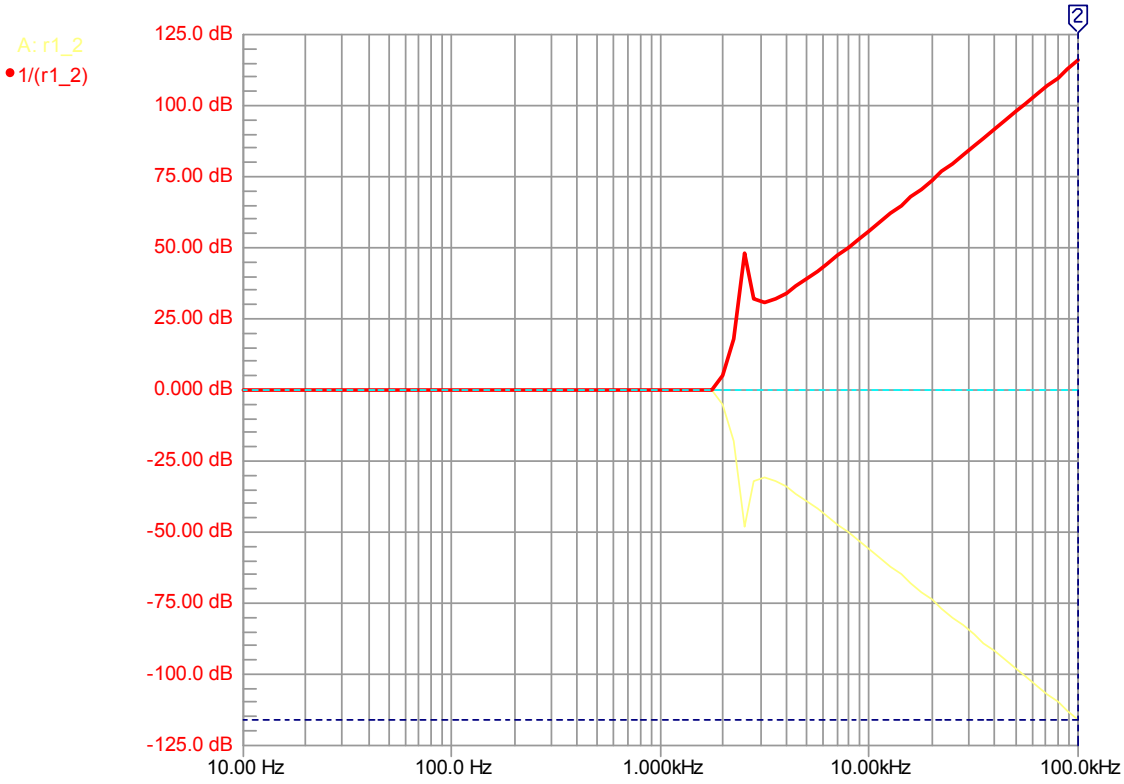
R=600 Ohm

f_c=2000 Hz

Una vez que hacemos los calculos correspondientes nos queda el filtro compuesto asi:



Nos interesa obtener la atenuación de este filtro, bueno la manera de hacerlo es muy sencilla. Obtenemos el diagrama de Bode del sistema , y con la funcion **math** calculamos la inversa del diagrama de bode que correspondera al gráfico de atenuación en dB



Measurement Cursors

1	1/(r1_2)	X: 1.0000	Y: 131.34n
2	r1_2	X: 100.00k	Y: -115.86

Maximum 1 . . 2 Requires 1 Wave

- Nótese que el pico que aparece un poco después de la frecuencia de corte corresponde a la frecuencia de atenuación infinita que luego decae y vuelve a subir, típica de los filtros compuestos debida a sus semisecciones L m-derivadas en cascada.