Laboratorio: Spice

Objetivos

Conocer el estándar de simulación de circuitos SPICE y realizar pequeñas simulaciones con el mismo.

SPICE es una forma elegante y sencilla de codificar circuitos eléctricos de manera que puedan ser procesados por un ordenador. Mediante un sencillo lenguaje podemos definir resistencias, fuentes de alimentación, etc., las conexiones entre ellos y los resultados que deseamos obtener.

Descripción

**SPICE** es una abreviabiación de *Simulation Program with Integrated Circtuit Emphasis*. Se trata básicamente de un método estándar para describir circuitos usando texto plano en lugar de una representación gráfica (o *esquemática*). A esta descripción en texto se la llama también ***netlist*** y básicamente se corresponde con la *lista* de los componentes del circuito y cómo estos están conectados entre sí, es decir, de los nodos de unión.

Los ficheros *netlist* pueden tener extensiones .cir, .net o .sp y es muy común encontrárselos con cualquiera de estas.

Existen en el mercado muchas variantes (intérpretes) de Spice, aunque el original fue descrito en la Universidad de Berkeley. En la lista de intérpretes de Spice tenemos desde esfuerzos y proyectos comerciales hasta *open source* y regidos por distintas comunidades de usuarios y programadores.

**Pregunta**: ¿qué comparación puedes efectuar entre C y Spice como estándares (lenguajes) y sus respectivas implementaciones en *software*? ¿Qué implementaciones reales (compiladores) del lenguaje C conoces?

**Elementos de un *netlist***

Como acabamos de comentar, un *netlist* se corresponde con la codificación de los elementos electrónicos de un circuito y las uniones entre los mismos. Veamos con más concreción qué partes y secciones lo componen.

**Comentarios**

La primera línea de un *netlist* se corresponderá siempre con un comentario. A partir de esta línea se pueden introducir más comentarios pero tienen que ir siempre precedidos de un \*. Ejemplo:

Mi primer circuito

\* Otro comentario

\* más comentarios

**Dispositivos**

Los elementos de un *netlist* son los mismos que encontramos en cualquier circuito eléctrico sencillo, tales como resistencias, **condensadores**, **bobinas**, **interruptores**, **hilos** y **fuentes** de alimentación. Para distinguir uno de otro, se reserva una letra característica: **V** para fuentes de alimentación, **R** para resistencias, **C** para condensadores y **L** para bobinas. También es posible usar estas letras en su versión en minúscula (**r**, **v**, **c**, **l**).

Después de esta letra característica se puede sufijar cualquier texto para diferenciar un elemento de otro (números, letras, palabras, etc.). Ejemplo:

\* Una Resistencia

R1

\* Otra Resistencia

R2

\* Fuente de alimentación

V

\* Un condensador

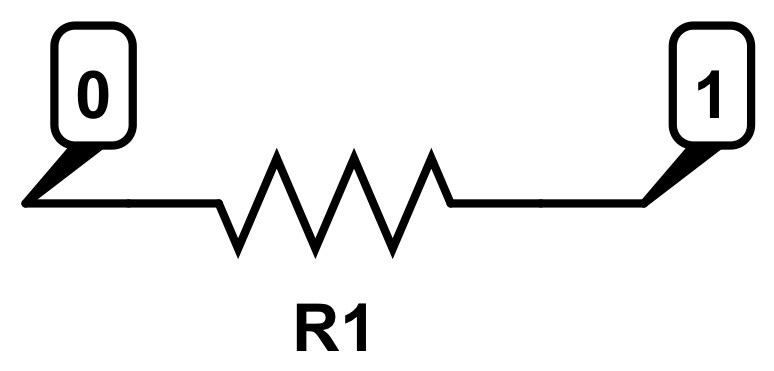
Cprincipal

**Conexiones**

A continuación de indicar el elemento eléctrico, tenemos que informar a Spice cuáles son los puntos de unión tanto a un lado como al otro del elemento. Así es como Spice sabe qué está conectado a qué: porque comparten un **punto** (o **nodo**, aunque este término se reserva sobre todo a uniones de más de dos elementos) que hemos señalizado correctamente. Para nombrar nodos, lo mejor es emplear una numeración secuencial: 0...n. **La enumeración de los puntos de unión es completamente a nuestro criterio**.

\* Una resistencia  
\* entre cables 0 y 1  
R1 0 1

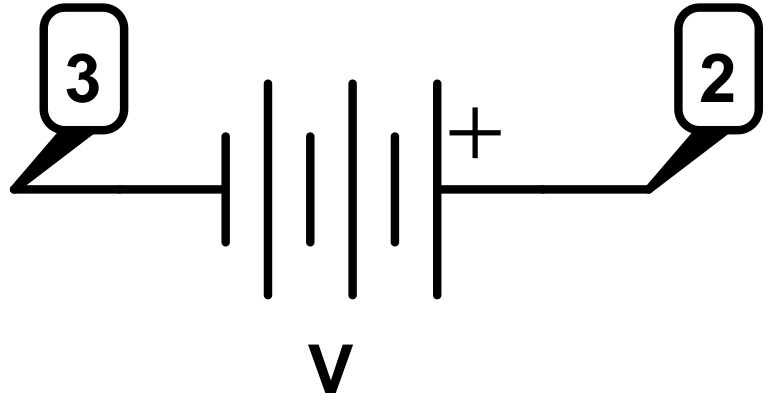
Que es equivalente a:



**Solo es necesario seguir un criterio**: en el caso de una  
fuente de alimentación, el nodo que pondremos primero será  
aquel que está más cerca del *borne* positivo.

\* Para una fuente indicamos primeramente conexión a nodo positivo.  
v 2 3

Que es equivalente a:



**Unidades en SPICE**

Las unidades de las magnitudes características del circuito son siempre unidades del sistema internacional y no es necesario indicarlo explícitamente en el *netlist*.

La forma de especificar múltiplos de estas cantidades es añadiendo una letra. Básicamente las que nos interesan y las que suelen aparecer mayoritariamente son k para «kilo-», m para «mili?» y u para «micro?». En el caso de las fuentes de alimentación hemos de especificar si se trata de corriente continua (dc) o alterna (ac).

\* Una resistencia de 5 Ohmios  
R2 1 0 5  
\* Una pila de 10 Voltios (continua)   
V1 1 0 dc 10  
\* Una resistencia de 5 kΩ  
RX 2 4 5k

**Pregunta**: ¿qué unidades del Sistema Internacional relacionadas con la asignatura –y los circuitos en general– conoces?

**Valores iniciales**

Aparecen justo al final de la definición del componente (ic). Suelen aplicarse principalmente con condensadores.

\* Una condensador inicialmente no cargado  
c 1 0 1u ic=0

**Fin del circuito**

El fin de la descripción de un *netlist* se especifica mediante el  
comando .end.

\* Mi primer circuito   
V 1 0 dc 10   
R 1 0 5  
\* Fin del circuito   
.end

**Comandos SPICE para circuitos en corriente continua**

Además de la descripción del circuito, hemos de indicar al intérprete de Spice qué tipo de análisis queremos realizar sobre el mismo y cómo queremos presentar la salida de la simulación. Los comandos en Spice empiezan por un «.» y suelen escribirse justo al final del circuito, pero antes del comando .end.

Mi primer circuito  
\* Aquí van los componentes   
R 1 0 6k  
...  
\* Comandos  
.op  
...  
\* Fin del circuito  
.end

**Pregunta**: hasta lo que has visto del lenguaje SPICE, ¿dentro de qué tipo o conjunto de lenguajes encajaría?, ¿funcionales?, ¿específicos de dominio?, ¿procedurales?, ¿estructurados?, ¿orientado a objetos?, ¿funcionales? Justifica tu respuesta.

En principio, Spice define también unos comandos para presentar datos en varios formatos, pero el intérprete de Spice que vamos a utilizar (LTspice) en este laboratorio se ocupa de manera automática (sin necesidad de que se lo pidamos explícitamente) de esta presentación de resultados.

Veamos los principales comandos de simulación:

* **.op** es el comando más sencillo que podemos emplear en SPICE. Devuelve el voltaje e intensidad en cada ramal y componente del circuito. Este comando no necesita parámetros.
* **.dc** es muy parecido al comando .op pero nos permite cambiar el valor del voltaje de una fuente de alimentación en pasos consecutivos entre el valor A y el valor B. En el caso de que la fuente tuviera asignada ya un valor para su voltaje, este sería ignorado. Ejemplo:

\* Variamos el valor del voltaje   
\* de la fuente v de 1 a 1000   
\* en pasos de 5 voltios  
v 1 0  
.dc v 1 1000 5  
v2a 2 4 dc 9  
\* Igual para v2a. Se ignora su voltaje de 9V  
.dc v2a 2 20 2

En el caso de LTspice, cuando encuentra una orden .dc, genera una gráfica directamente sin necesidad de solicitarla mediante el comando .plot, que veremos a continuación.

El comando **.tran** realiza un análisis en el tiempo de los parámetros del circuito. Si no se emplea la directiva UIC, este análisis se realiza desde el punto estable de funcionamiento del circuito hasta un tiempo tfinal. y en intervalos tstep. Si empleamos el parámetro UIC (*Use Initial Conditions*), entonces se hará uso de las condiciones iniciales definidas para cada componente (típicamente ic=X en el caso de los condensadores).

\* Hacemos avanzar el tiempo entre  
\* tinicial y tfinal en pasos tstep  
.tran tstep tfinal [tinicial] [UIC]  
\* Lo más normal es que solo indiquemos tfinal   
.tran 2

En el caso de LTSpice, al procesar un comando .trans, se nos  
genera automáticamente una gráfica de manera automática. En otros entornos Spice tenemos que solicitar la gráfica explícitamente mediante el comando .plot.

**Introducción a LTspice**

LTSpice es un intérprete de Spice de libre distribución (aunque no de código abierto) realizado por la empresa de componentes electrónicos Linear.

LTspice es tanto un intérprete de Spice como un programa que nos permite diseñar circuitos de manera gráfica, sin embargo, vamos a reservar esta parte para más adelante. Por ahora vamos a hacer uso únicamente de su *cerebro simulador* y le vamos a facilitar como *input* ficheros *netlist* creados por nosotros.

En el próximo ejemplo vamos a simular un circuito en corriente continua pero al que le vamos a aumentar el valor de la fuente de alimentación desde 0 a 1 000 voltios en pasos consecutivos de 5 voltios. Lo primero que tenemos que hacer es importar el archivo en el que hayamos escrito el código *netlist* de este circuito en el LTspice. En la versión de Windows de LTspice no podemos iniciar el **editor de *netlist*** (que no es otra cosa que un sencillo editor de texto), pero sí podemos abrir un circuito creado con cualquier otro editor de texto, incluso un documento vacío pero con la extensión apropiada. Sin embargo, en la **versión de macOS** sí podemos crear un fichero vacío de texto donde empezaremos a definir nuestro *netlist*. En la versión de Windows podemos hacer también uso del menú contextual del explorador.

Una vez creado o abierto el fichero *netlist*, LTspice reconocerá  
el formato y aplicará inmediatamente la sintaxis de color apropiada.

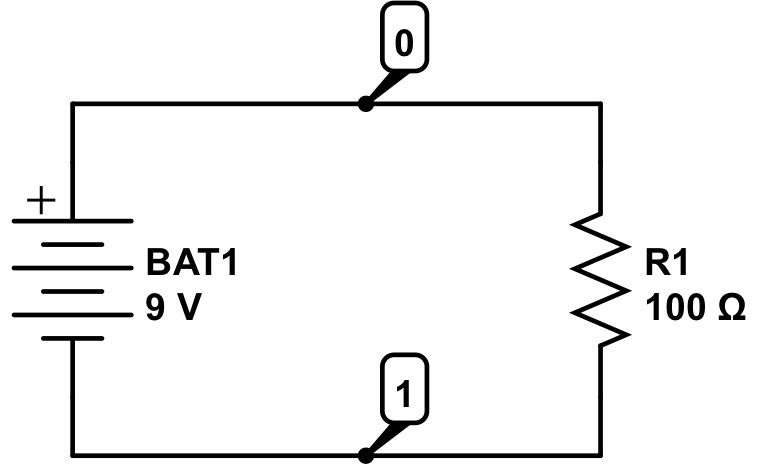
**Pregunta**: ¿cómo consigue LTspice realizar una sintaxis  
de color apropiada? ¿Cómo se llama al proceso que realiza  
la identificación correcta de los distintos componentes de  
un lenguaje de programación?

**Simulaciones a efectuar**

Realiza las siguientes simulaciones.

**Primeras simulaciones: ley de Ohm**

Vamos empezar con una simulación sencilla que va a consistir en la aplicación directa de la ley de Ohm. Para ello, crea la representación *netlist* de este sencillísimo circuito:

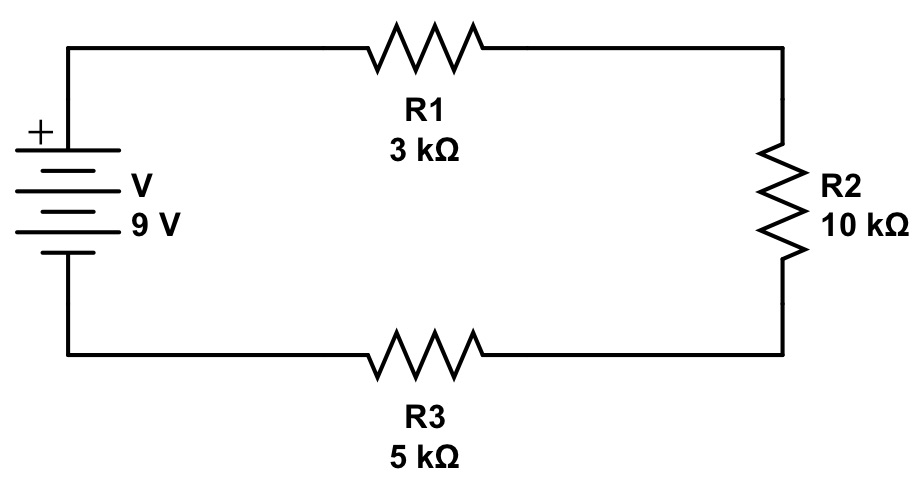


Además de establecer correctamente los componentes del circuito, tienes que especificar qué análisis quieres hacer sobre él. En este caso será de tipo .op.

Para lanzar la simulación tan solo tenemos que pulsar el botón. En el caso de la versión de Windows veremos una ventana donde podremos ver los resultados en modo texto, pero en el caso de la versión de macOS, tendremos que usar la vista de gráfica (que también se abrirá automáticamente y es igual de informativa que la vista de texto). Si el resultado es una única cifra numérica (intensidad en una resistencia, voltaje en un nodo, etc.) veremos una línea recta de pendiente nula y con una ordenada en origen correspondiente al valor que buscamos.

Además, en el visor gráfico es necesario indicar manualmente qué gráficas queremos ver. Básicamente tenemos que añadir nuevas «trazas» a la gráfica y  
elegir aquella que nos interese en cada momento. LTspice nos presentará aquellas para las que ha calculado valores.

Realiza también la simulación .op del circuito siguiente:



**Segunda simulación: ley de Ohm para valores discretos y crecientes del voltaje**

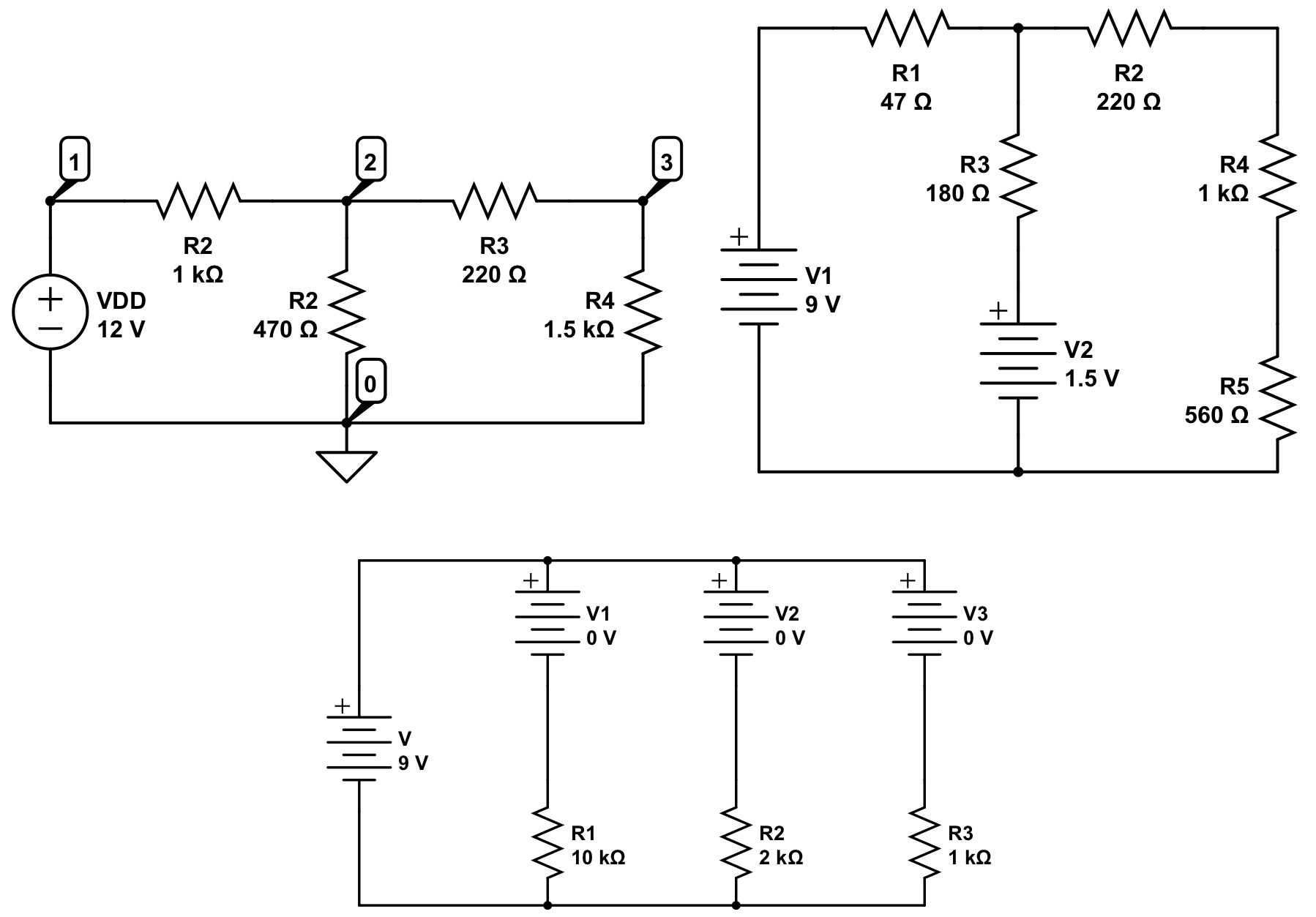
Utiliza el comando apropiado de Spice (pista: .dc) para obtener varios valores consecutivos de la intensidad que circula por la resistencia del circuito anterior. Represéntalos gráficamente.

**Tercera simulación: resistencias en serie**

Obtén el *netlist* de este circuito y halla la caída de tensión entre los bornes de la resistencia R2 de circuito anterior.

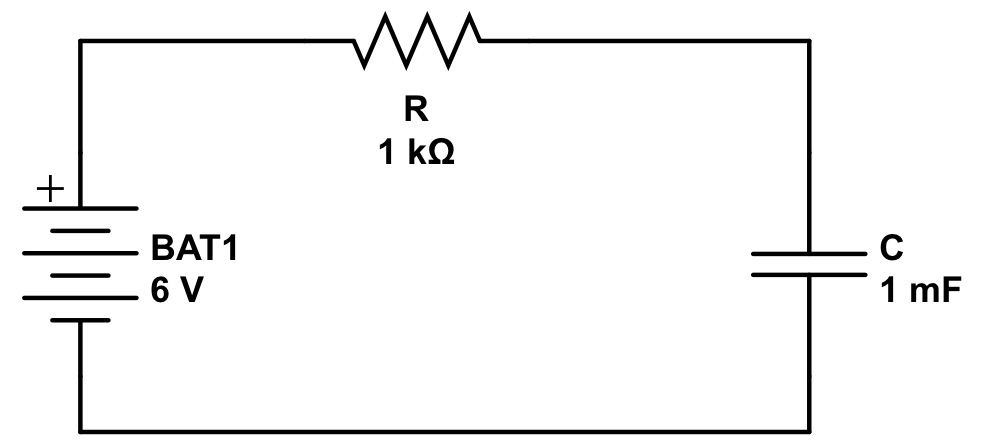
**Cuarta simulación: resistencias en paralelo**

Crea el fichero *netlist* de los siguientes circuitos (se han señalado los puntos de unión en el primero a efectos de ayuda). Obtén las intensidades y corrientes en cada malla y resistencia.



**Quinta simulación: carga de un condensador**

Obtén el *netlist* de este circuito y, tras simularlo, haya el momento (en segundos) que tarda el condensador en cargarse completamente. Pista: tienes que emplear un análisis en el tiempo del circuito (.trans). Asume que la corriente inicial (t=0) que pasa por el condensador es nula.



**Sexta simulación: circuito en corriente alterna**

Realiza a continuación una sencilla simulación de un circuito con fem sinusoidal. La simulación consistirá en un análisis en el tiempo de la intensidad de corriente presente en cada instante en la resistencia entre 0 y 30 milisegundos. ¿Se hace nula en algún momento? Si es así, ¿en qué momentos es nula?

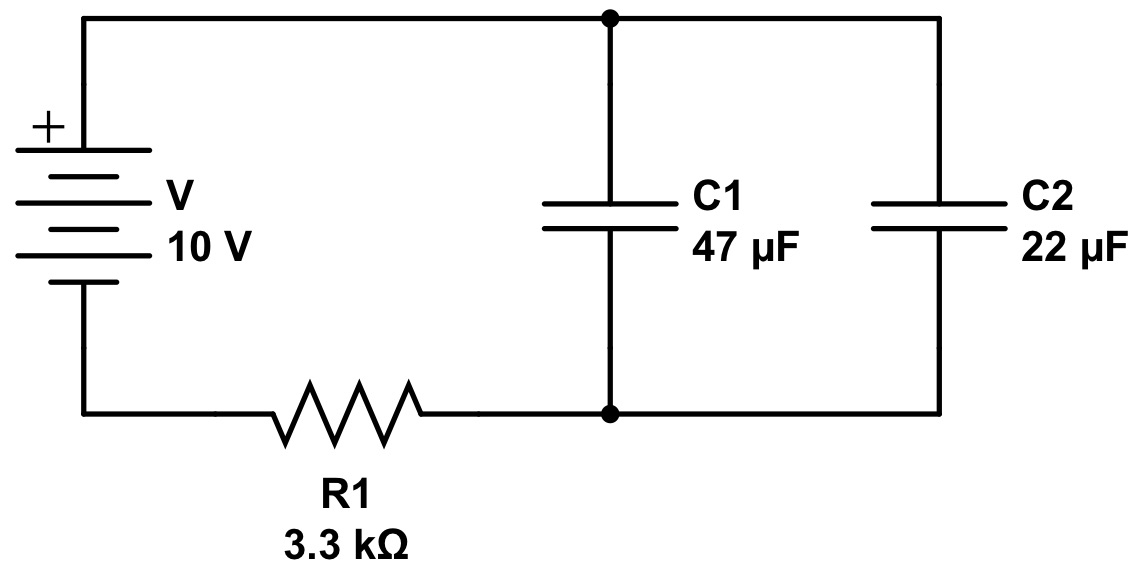
Para definir en el *netlist* una fuente alterna ideal que siga una función periódica determinada, tenemos que sustituir el argumento dc por:

función(voffset amplitud frecuencia retardo amortiguación fase)

Supón para el ejercicio que el *offset* (desplazamiento en origen), el retardo y fase son nulos.

**Séptima simulación: condensadores en paralelo en corriente continua**

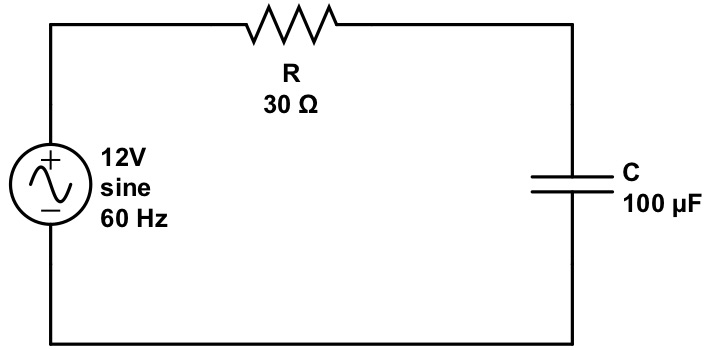
Realiza a continuación la descripción en *netlist* y simulación de este circuito que consiste en la asociación de condensadores en paralelo. ¿Qué rama se cortocircuita antes: la de C1 ó C2? Asume que ambos condensadores no están cargados inicialmente.



**Octava simulación: circuito en corriente alterna**

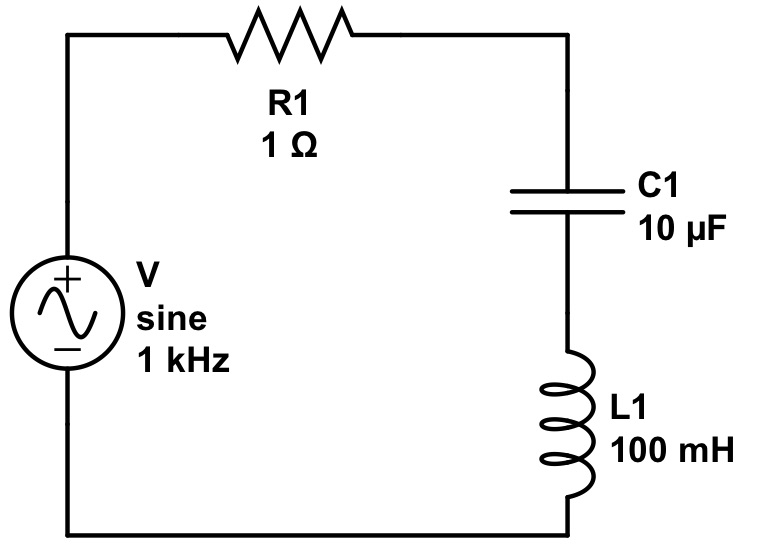
Diseña el circuito de la figura que consiste básicamente en una fuente sinusoidal, una resistencia y un condensador. Representa gráficamente la evolución en el tiempo de la corriente que pasa por la resistencia del circuito en los 3 segundos iniciales.

Recuerda que tienes primero que iniciar la simulación y añadir trazas en el visor de gráficos. También se pueden añadir trazas mediante el ratón si nos aproximamos a un cable o elemento del circuito. En este caso aparecerán los cursores y que permiten añadir trazas de voltaje e intensidad de corriente sobre el elemento/cable elegido.



**Novena simulación: frecuencia de resonancia**

Diseña el circuito de la figura que consiste básicamente en una fuente sinusoidal, una resistencia, un condensador y una bobina. Se trata de un típico circuito RLC típico con una frecuencia de resonancia específica. Para esta frecuencia, la intensidad del circuito es máxima.



Para descubrir cuál es esta frecuencia, en principio deberíamos ir variando a mano la frecuencia de la fuente sinusoidal y ejecutando una nueva simulación en cada iteración. Sin embargo, en Spice contamos con el comando .ac que nos permite variar la frecuencia de una fuente de manera automática, consecutiva y en intervalos definidos.

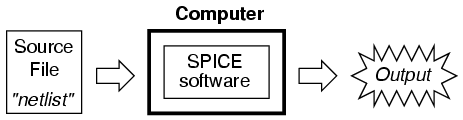
Para hacer este tipo de análisis, a la hora de definir la fuente sinusoidal tenemos que usar el apartado de análisis AC del cuadro de parámetros como indica la figura. A continuación hemos de añadir al circuito el comando de análisis:

.ac lin 20 100 200

Esta instrucción informa al intérprete de Spice que queremos variar la frecuencia desde 100 a 200 Hz en pasos de 20 Hz. Ejecuta el análisis y representa el valor de la intensidad que circula por la resistencia. LTspice graficará este valor en función de la frecuencia automáticamente porque es suficientemente inteligente para saber que se quiere hacer un análisis de frecuencia.

**Python y Spice con Ahkab**

Como se ha comentado en la parte de presentación del laboratorio, **Spice** es un estándar que puede tener multitud de implementaciones.



Como no podía ser de otra manera, existen también algunas en Python. Las principales son:

* PySPICE (<https://github.com/rca/PySPICE>) que no es más que un *wrapper* entorno a CSPICE (<https://naif.jpl.nasa.gov/naif/>), de la NASA.
* Ahkab (<https://github.com/ahkab/ahkab>), escrito completamente en Python y el que vamos a utilizar.

**Instalación de Ahkab**

Para instalar Ahkab, simplemente tenemos que hacer uso del gestor de paquetes para Python pip. Puedes usar tanto Anaconda como Cygwin, o incluso el Python nativo de tu sistema operativo.

$ pip install ahkab

A partir de este momento, bien podemos invocar Ahkab como *framework* o como programa independiente (intérprete Spice). Por ejemplo, podemos ejecutar el análisis y circuito descrito en un fichero *netlist* de la siguiente manera: hola

$ ahkab circuito.net

Ahkab analizará el circuito y presentará en el *shell* los resultados del análisis.

**Ahkab y los *netlist***

Ahkab es capaz de leer *netlists* sin mayor problema, sin embargo, como en toda implementación de un lenguaje, existen *cismas* y cada *facción* tiene su propia versión diferente.

**Pregunta**: enumera otros cismas en la historia reciente de la informática.

La única diferencia significativa es la definición de fuentes de alimentación. En Ahkab es necesario especificar si la fuente es de señal continua o no. Por ejemplo, en el siguiente ejemplo tenemos dos fuentes de alimentación DC de 1 y 2 voltios respectivamente.

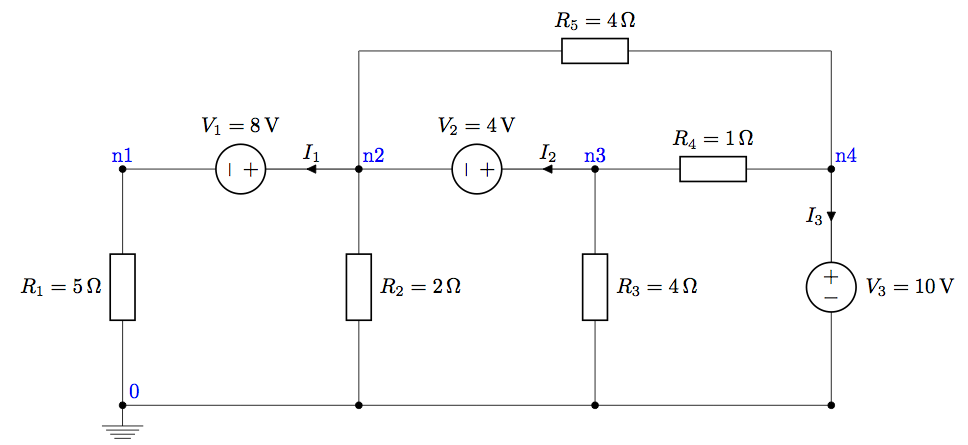
\* Circuito sencillo  
v2 0 1 vdc=1  
r2 2 0 2  
v1 0 2 vdc=2  
r1 0 1 1   
.op  
.end

Tienes un montón de ejemplos de *netlist* en el código fuente (<https://github.com/ahkab/ahkab>) de Ahkab, concretamente en sus carpetas doc y tests. La única diferencia es que en Ahkab, los ficheros *netlist* tienen extensión .ckt ó .spc en lugar de .net. La extensión es lo de menos.

Ahkab también tiene un recopilatorio muy bueno (<http://ahkab.readthedocs.org/en/latest/help/Netlist-Syntax.html>) sobre la sintaxis del lenguaje Spice.

**Un circuito sencillo como ejercicio**

Sigue las instrucciones que vienen en el tutorial (<https://ahkab.readthedocs.io/en/latest/examples/Simple_OP.html>) de Ahkab y reproduce el ejemplo. Se trata de un sencillo análisis OP.



import ahkab  
from ahkab import new\_ac, run  
from ahkab.circuit import Circuit  
from ahkab.plotting import plot\_results   
import numpy as np  
mycir = ahkab.Circuit('Simple Example Circuit')  
mycir.add\_resistor('R1', 'n1', mycir.gnd, value=5)  
mycir.add\_vsource('V1', 'n2', 'n1', dc\_value=8)  
mycir.add\_re V3', 'n4', mycir.gnd, dc\_value=10)  
mycir.add\_resistor('R5', 'n2', 'n4', value=4)  
opa = ahkab.new\_op()  
r = ahkab.run(mycir, opa)['op']  
print(r)

Repite el ejemplo (incluido el análisis .op), pero leyendo el mismo circuito desde un fichero *netlist* externo y empleando Ahkab como intérprete Spice. Tendrás que traducir el circuito a este tipo de ficheros como los trabajados hasta ahora.

**Análisis AC con Ahkab y Gnuplot**

Vuelve al circuito que vimos en el ejercicio curva de carga de un condensador y rehazlo en código Python+Ahkab.

Luego, ejecuta un análisis AC (.ac) (<https://ahkab.readthedocs.io/en/latest/ac.html>) desde Python y representa la gráfica siguiendo el ejemplo (<https://github.com/ahkab/ahkab>) de la documentación de Ahkab. Un ejemplo:

# análisis AC empezando en 0, hasta 1, en pasos de 0.1  
# new\_ac(inicio, fin, numero\_de\_puntos\_a\_pintar, x0=None)  
ac = new\_ac(0, 1, 0.1, x0=None)  
# Lo ejecutamos  
res = run(cir, ac1)  
# Dibujamos los resultados  
plot\_results('Condensador', [('','')], res['ac'], outfilename='condensador.png')

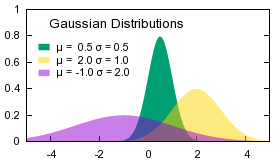
Ejecuta el mismo análisis desde el *netlist* que compusiste, pero ahora usa Ahkab como un programa de *shell*:

$ ahkab carga-condensador.net

Deberías ver por pantalla un listado de datos como el siguiente:

#T V1 V2 I(V1)  
1.000100010001000106e-03 6.000000000000000000e+00 5.994604855623937752e-03 -5.994005401150376104e-03  
2.000200020002000211e-03 6.000000000000000000e+00 1.198621360725866253e-02 -5.988013792386741654e-03  
2.208472586375065643e-03 6.000000000000000000e+00 1.323322274712842271e-02 -5.986766783258871093e-03

Lo que ocurre es que cuando Ahkab funciona así (en *modo intérprete*), no guarda ningún dibujo sino que simplemente *escupe* resultados. Para dibujarlo, guarda los datos en un fichero de texto y usa el software de graficado Gnuplot (<http://www.gnuplot.info/>). Puedes descargar binarios para Mac y Windows. En el caso de macOS, te recomiendo Homebrew para instalar Gnuplot.



Para dibujar con Gnuplot, puedes seguir este pequeño ejemplo, pero te sugiero que leas algo de la documentación ([http://www.tatsuromatsuoka.com/gnuplot/Eng/winbin/http://www.gnuplot.info/documentation.html](http://www.tatsuromatsuoka.com/gnuplot/Eng/winbin/http:/www.gnuplot.info/documentation.html)) de este *milenario* y asiduo programa en ciencia.

# redirijimos salida de ahkab a un fichero en lugar de visualizarla por pantalla  
$ ahkab carga-condensador.net > carga-condensador.tsv  
$ gnuplot   
# pintamos la 4ta columna, con la intensidad  
gnuplot> plot 'carga-condensador.tsv' using 4

Entrega todo el código, *netlists* y gráficos que se piden. Para guardar el gráfico de Gnuplot **no hagas una captura de pantalla**, sino que guarda la información representada usando otro terminal, como PDF (<http://gnuplot.sourceforge.net/docs_4.2/node407.html>).

Rúbrica

* Realizar correctamente las simulaciones (peso 35%).
* Responder correctamente a las preguntas que se proponen (peso 35%).
* Correcta presentación y narración digna de un nivel universitario (peso 30%).

Extensión máxima de la actividad: 5 páginas como mínimo.