README 'themeltspice.sh'

> **WHAT IS THIS?** `'themeltspice.sh'` is a color theme manager for the **OSX** version of LTSpice electric circuit simulator which is a version of the classical SPICE (Simulation Program with Integrated Circuit Emphasis) simulator.
> LTSpice is made freely available by Analog Devices.

Quckstart <TL;DR>

1. Copy the file "`themeltspice.sh`" to a directory of your choice (or somewhere in your PATH). Change to that directory.
2. To list all available themes run:
   `$ ./themeltspice.sh -l`
3. To set a new theme, say '`darcula`', run:
   `$ ./themeltspice.sh darcula`

When you now run **LTSpice** you will see that the color palette has changed. You can see examples of all available themes as well.

That is all there is to it for the most basic use case. Read on if you want to know more!

Content

Introduction

|back to content table|

This is not meant to be an introduction to either the usage or function of the electric circuit simulator **LTSpice**. It is therefore assumed you have installed and have basic knowledge of the **OSX** version of **LTSpice**.

Note: This theme manager is unique to **OSX** and will not in any shape or form work on a the Window version of **LTSpice**. Instead see Windows **LTSpice** theme manager if you are looking for a Windows based **LTSpice** theme manager.

This script is used to create and set a color theme for the **OSX** version of **LTSpice**. The themes are stored as a plain text file in an human readable format (see the BNF grammar at the end of this README file).

While **LTSpice** does not inherently support any concept of a color theme the color settings are stored in an **OSX** standard property list (plist) configuration file that can be updated outside the **LTSpice** program. To avoid race conditions the **LTSpice** application needs to be closed when a theme change is made. This is checked by this script and if a running process is found an error message will be shown and the script terminates.

A note on the OSX version of LTSpice:

While much or all of the core funtionality of the simulator are exctly the same between the **OSX** and Window version the UI is dramatically different. In fact, many **OSX** users are so stumped by the apparent frugality of the **OSX** UI that they end up using the Windows version even on **OSX** by running it under Wine. This is a mistake (but perhaps understandable if only barely).

While the **OSX** version does not adhere to the ususal design guidelines for **OSX** programs and requires some "getting used to" it is a highly functional UI for its purpose. After the initial "getting-used-to" experience many users will hopefully realize that the **OSX** version is superior for professionals (or even serious amateurs) compared to the window version. This is mainly to do with the abondonment of menus that distract the user and forces eye-focus to shift.

Both the advantage and the drawback of the **OSX** UI is that it heavily relies on 1) the user getting familiar and learning a few important shortcut keys and 2) becoming familiar with context sensitive menus.

Once those keystrokes are mastered it is usually substantially faster to create a circuit diagram and setup a simulation in the native **OSX** version than the Windows dito. The **OSX** version also have context sensitive menu (trackpad "right-click") in most places.

To be fair. The **OSX** version does have some missing functionality but nothing really serious for pro or semi-pro usage. The main *functional* differences are:

1. No dialogue help to enter `.meas` simulation command.
2. No keyboard shortcut editor
3. Not possible to edit `.op` operation point labels to, for example, change from the default voltage display to current through an element or perhaps change the number of decimals shown in the diagram on an `.op` label.

Why do this as a bash shell script?

Why oh why was this done as a bash shell script I can hear people cry out. Couldn't it be written in [select favourite language] (e.g. Python). Of course it could. However, bash is the lowest common denominator that doesn't require any dependencies and the guiding principle of this has been that it should run out of the box. Using a self-contained shell script is an easy way to avoid the potential *"module/version-hell"* of Python. Instead we claim it is perfectly possible to write readable, medium-complex programs using bash. It is of course not without its limitation since bash code can be almost unreadable when one uses all of the features available that are not commonly well known. If you stick to some good design principles (and modularization) it is perfectly readable and maintainable. Just like any language! If you envision a program with more than around 600-800 lines of manually written code then bash might not be your first choice. Especially not for the very old version of bash that default ships with **OSX** (v3.2.57). A lot has happened since that version was release well over a decade ago.

So why not write it as a `zsh` script? It would be perfectly fine to convert the few bash:ism used to `zsh` (since `zsh` even has a "bash" compatibilty mode) and it might very well be a good idea to pursuite.

Related work

The inspiration for this work comes from the Windows **LTSpice** theme manager. While this implementation is widely different in both function, form and implementation the drive to write this came out of friendly *"jealousy"* that the windows world had this but not the **OSX** world. That state of affairs cannot stand and has now been corrected!

Installation

|back to content table|

There is no installation program for this since it is only one executable script file and you are free to place that file anywhere at your conveniance.

To use the script either copy the script (`themeltspice.sh`) to some standard location for scripts as per your `PATH` variable or create a new directory and copy the file there and run it from this directory.

The script uses the default location of "`~/.ltspice_themes`" to store the theme file as well as a backup file of **LTSpice** original plst file when you first run the theme script. If the directory does not exist it will be created the first time you run the script. If no theme file exist a default theme file with six themes will be automatcally installed.

The default theme file is named "`themes.ltt`". The file-extension of this can be read as "**LT**Spice **T**hemes". By using the "`-f`" option you can also specify another file location to be used a theme file.

This default theme file installed (as of this writing) contain these six themes:

1. default (LTSpice default)
2. sakabug
3. twilight-after-dawn
4. dracula
5. softdark
6. blackwhite

Themes no 2-4 are taken directly from the Windows **LTSpice** theme manager. The theme "softdark" is an additional dfferent theme I personally like to use. The last theme "blackwhite" is especially suitable when printing a circuit diagram.

Later on if you find themes you like somewhere else just open the theme file and copy them at the end with one blank line between the new theme and the last existing theme.

You can easily check if the new theme have been added correctly by listing the theme with the command:

```
$> themeltspice.sh -l
```

> **WARNING!** The format of the **OSX** and Windows version of the theme files are not compatible since the developer of **LTSpice** have used different names for the control fields in the Windows and the **OSX** version.

Known Limitations

- It is not (yet) possible to remove or overwrite an existing theme. For that you have to first manually delete the theme you want to overwrite in the themes file.
- Windows theme files are not compatible with **OSX** and vice versa. *C'est la vie!*

Usage

|back to content table|

```
Set or create a named color theme for LTSpice
Usage:
%themeltspice.sh [-f <FILE>] [-d] [-l] [-h] [<THEME>]
-d          : Dump current plist to default or named theme file as
specified theme
-f <FILE>   : Use the specified file as theme file
-h          : Print help and exit
-l [<NAME>] : List themes in default or named theme file or íf <NAME>
              is specified check if <NAME> theme exists
-p          : List content in **LTSpice** plist file
-q          : Quiet no status output
```

There are two major use cases:

1. Set a new theme to be used the next time **LTSpice** is run
2. Save an existing color configuration you have made as a new theme

In addition to these major use cases there are some supporting function that are available

- List the names of all defined themes in a specific themes file
- Check if a specified theme esists
- Print the **LTSpice** binary configuration file in a human readable format

> **Note:** A copy fo the original **LTSpice** plist configuration file is also stored in the theme directory with the extension "`*.ORIGINAL`". In case (for some reason) the configuration file gest corrupt you can always restore a clean backup.

Setting a new theme for LTSpice

If we assume you have installed the script somewhere in your path (see Installation section above) you can now set the '`softdark`' theme as so:

```
$> themeltspice.sh softdark
Successfully updated new theme to 'softdark'
$> _
```

This will update the current **LTSpice** configuration file with this color schema. If you now start **LTSpice** you will see the effect of this theme switch. To restore back to the default schema just do:

```
$> themeltspice.sh default
Successfully updated new theme to 'default'
$> _
```

and there is nothing more to it. The settings are done in an atomic way so a change go through successfull or not at all. This way you cannot end up with a half-updated configuration file.

> **Atomic update:** The way this is done is by first copying the config file to a temporary directory, do the changes, run a integrity verification on the config file and then copy it back to the application location (i.e. `/Users/<USER>/Library/Preferences/com.analog.LTspice.App.plist)`)

Storing the current configuration as a theme

By first creating a color schema in **LTSpice** it can then be saved as a new theme. So if you want to store your current settings as the new theme, say, "`mytheme`" you use the "`-d`" (=dump) option as such

```
$> themeltspice.sh -d mytheme
Dumping current color setup from 'com.analog.LTspice.App.plist' to
'/Users/<USER>/.ltspice_themes/themes.txt' as theme 'mytheme'
$> _
```

This will store the new theme at the end of the existing theme file. If a theme with this name already exsts an error message will be printed informing about this.

> **WARNING:** You might have to quit and start **LTSpice** twice to force the update of the plist file from the plist cache before running the dump command.
> You might want to check that changes have been made by printing out the property list using the command:
> `$ themeltspice.sh -p`" (See Printing all settings)

Listing all themes available

To see a list of all themes defined use the "`-l`" option as so:

```
$> themeltspice.sh -l
Listing themes in '/Users/<USER>/.ltspice_themes/themes.txt''
 1. default
 2. sakabug
 3. twilight-after-dawn
```

```
 4. dracula
 5. softdark
 6. blackwhite
$> _
```

Checking if a named theme exists

Use the '`-l`' option with a theme name

```
$> themeltspice.sh -l mytheme
*** ERROR *** Theme 'mytheme' DOESN'T exists in
'/Users/<USER>/.ltspice_themes/themes.ltt'
```

or

```
$> themeltspice.sh -l default
Theme 'default' exists in '/Users/<USER>/.ltspice_themes/themes.ltt'
```

Printing all settings stored in the config file

To see the complete configuration file (and not only the color settings) use the "`-p`" (=print) that will dump **LTSpice** full property list file to stdout as in:

```
$> themeltspice.sh -p
'/Users/<USER>/Library/Preferences/com.analog.LTspice.App.plist' content:{
  "AllowShortedCompPins" => 0
  "AutoDotRawDeletion" => 1

  ...

  "WaveColor11" => 12237492
  "WaveColor12" => 255
  "WaveColor13" => 16748287
}
$> _
```

How the script works

|back to content table|

The configuration file where the **LTSpice** configurations are stored is a binary configuration file and cannot be directly manipulated. The format used is a standard **OSX** *"Property List"* (plist) and as such **OSX** provides a command line tool that can be used to manipuate the individual fields in that property file.

The **OSX** utility is called '`plutil`' and is used to read and manipulate individual fields in this configuration file. See `man plutil` for more details.

By default the plist configuration file of **LTSpice** is stored at:

```
/Users/<USER>/Library/Preferences/com.analog.LTspice.App.plist
```

Since **LTSpice** updates this file at least on every exit the simulator must be closed before running this script. This is also checked in the script and an error message is shown if any running copies are detected.

As an extra precaution the first time the script is run it creates a backup copy of the configuration file and stores it in the theme directory with the added suffix "`.ORIGINAL`".

Since **OSX** caches all plist files it is not enough to just update the property file on its own, one must also force a refresh of the property cash using "`default read <PLIST-FILE>`" command.

Directories and files used

- [`/User/<USER>/.ltspice_themes/themes.ltt`]
  The default location of themes
- [`/User/<USER>/.ltspice_themes/com.analog.LTspice.App.plist.ORIGINAL`]
  Copy of the **LTSpice** application plist file at the time of first run of this script
- [`/Users/<USER>/Library/Preferences/com.analog.LTspice.App.plist`]
  LTSpice application plist file

Theme file format

|back to content table|

For each theme a total of 34 color parameters are stored as listed in figure 1 below.

> **Difference between OSX the Windows version theme format:**
> Unfortunately there are three minor differences between the Windows version and **OSX** version theme files in that some fields have different names in the two versions.
> There are three major differences:
> --OSX uses the name "`GridColor`" while the windows version simply call it "`Grid`".
> --OSX uses the name "`InActiveAxisColor`" while the windows version simply call it "`InActiveAxis`".
> --The window version have a "`SchematicColor13`" which doesn't exist in the **OSX** version.
> For this reason it is not possible to copy theme directly between the Windows and **OSX** without som manual fixes.

```
GridColor
InActiveAxisColor
WaveColor0
WaveColor1
WaveColor2
WaveColor3
WaveColor4
WaveColor5
WaveColor6
WaveColor7
WaveColor8
WaveColor9
WaveColor10
WaveColor11
WaveColor12
WaveColor13
SchematicColor0
SchematicColor1
SchematicColor2
SchematicColor3
SchematicColor4
SchematicColor5
SchematicColor6
SchematicColor7
SchematicColor8
SchematicColor9
SchematicColor10
SchematicColor11
SchematicColor12
NetlistEditorColor0
NetlistEditorColor1
NetlistEditorColor2
```

```
  NetlistEditorColor3
  NetlistEditorColor4)
```

Fig 1: The fields stored as a color theme

BNF Grammar

The BNF grammar for the theme file is extremely simple and is shown in Figure 2. below

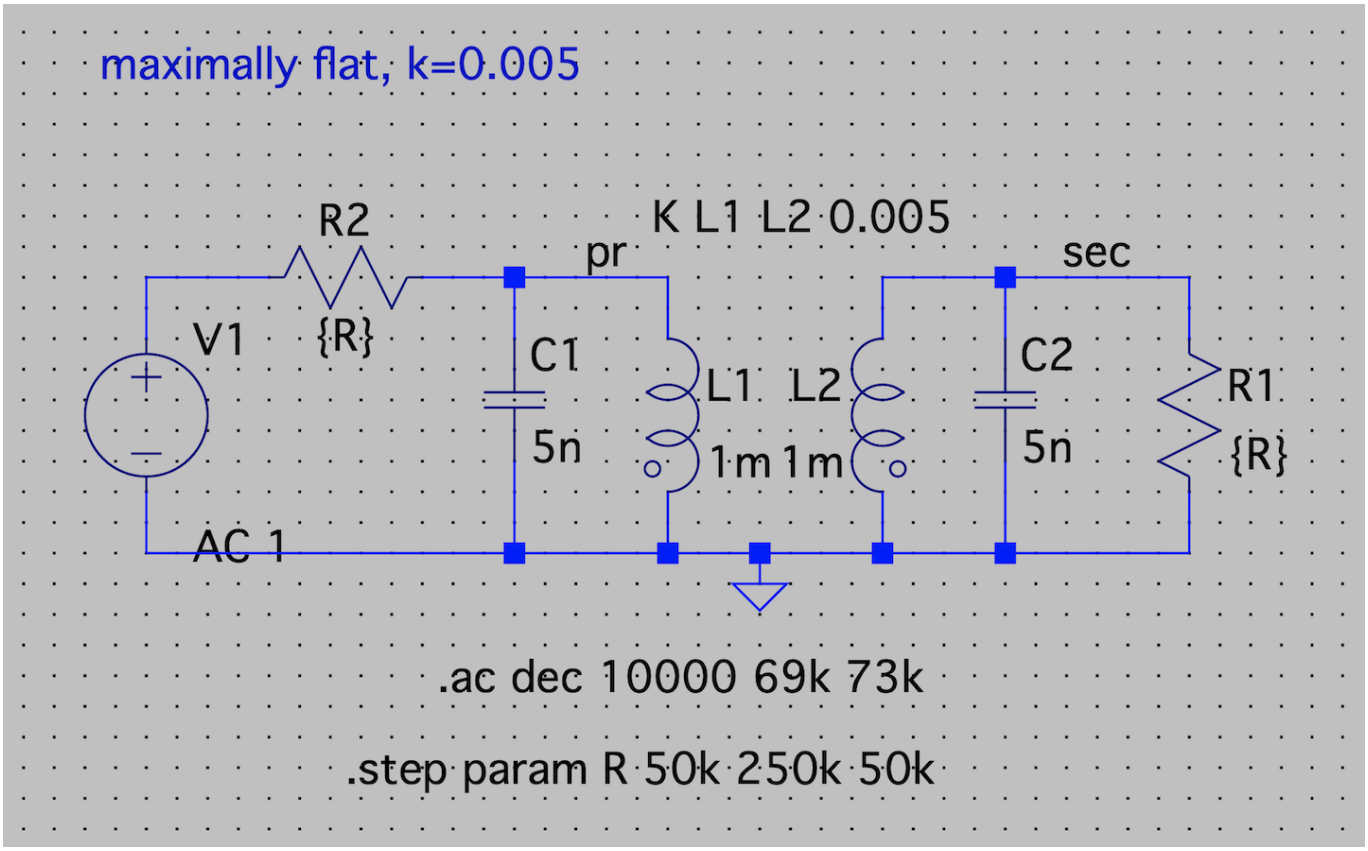
```
themes    ::= theme | theme <EMPTY_LINE> themes
theme     ::= '[' theme-name ']' <NL> fields
fields    ::= field | field <NL> fields
field     ::= fieldname '=' digits
fieldname ::= "" | alnum fieldname
alnum     ::= "A" | "B" | ...
digits    ::= "0" | "1" | ...
```

Fig 2: The simplified BNF grammar for the themes file format

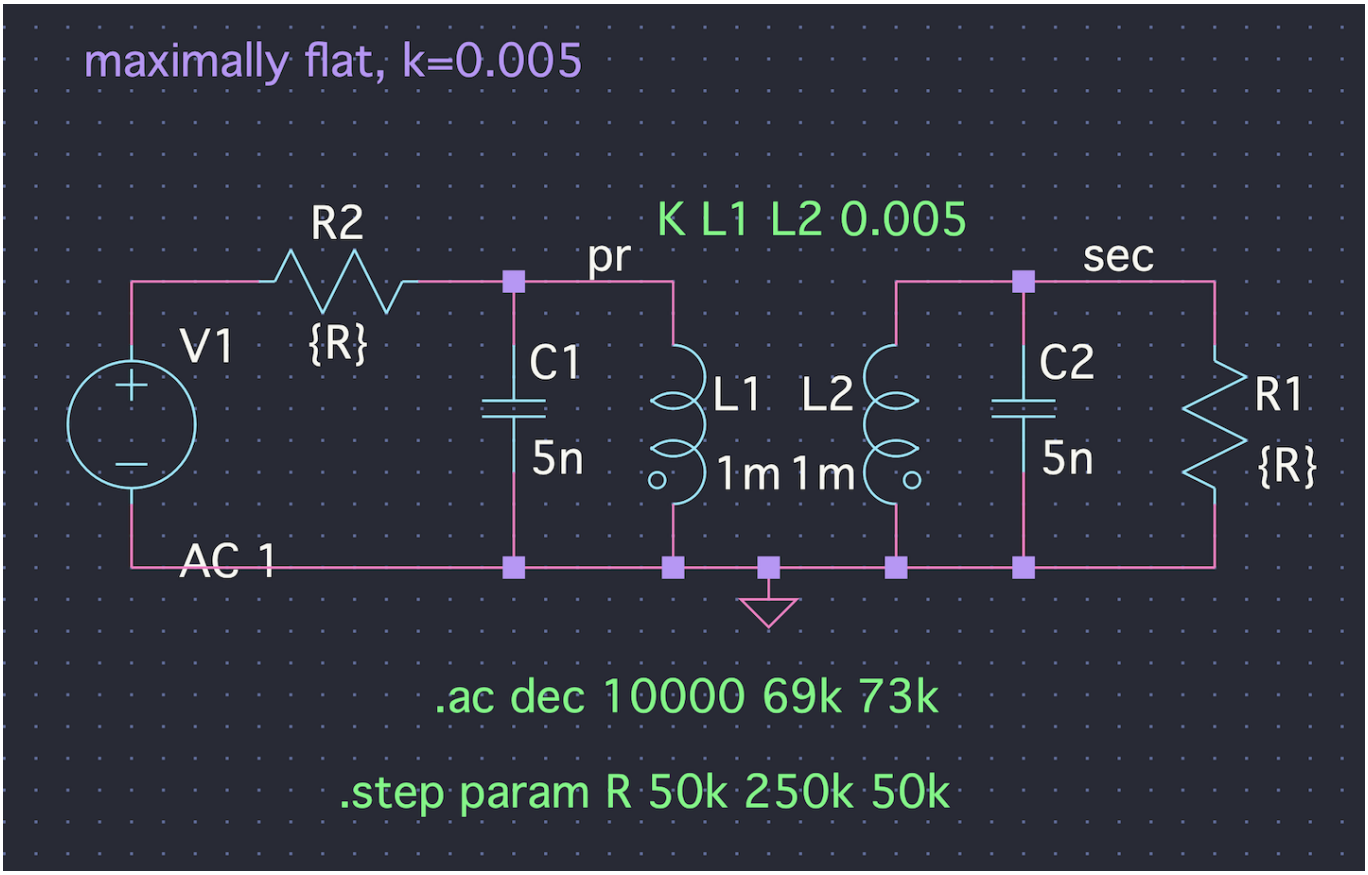
Theme reference screenshots

|back to content table|

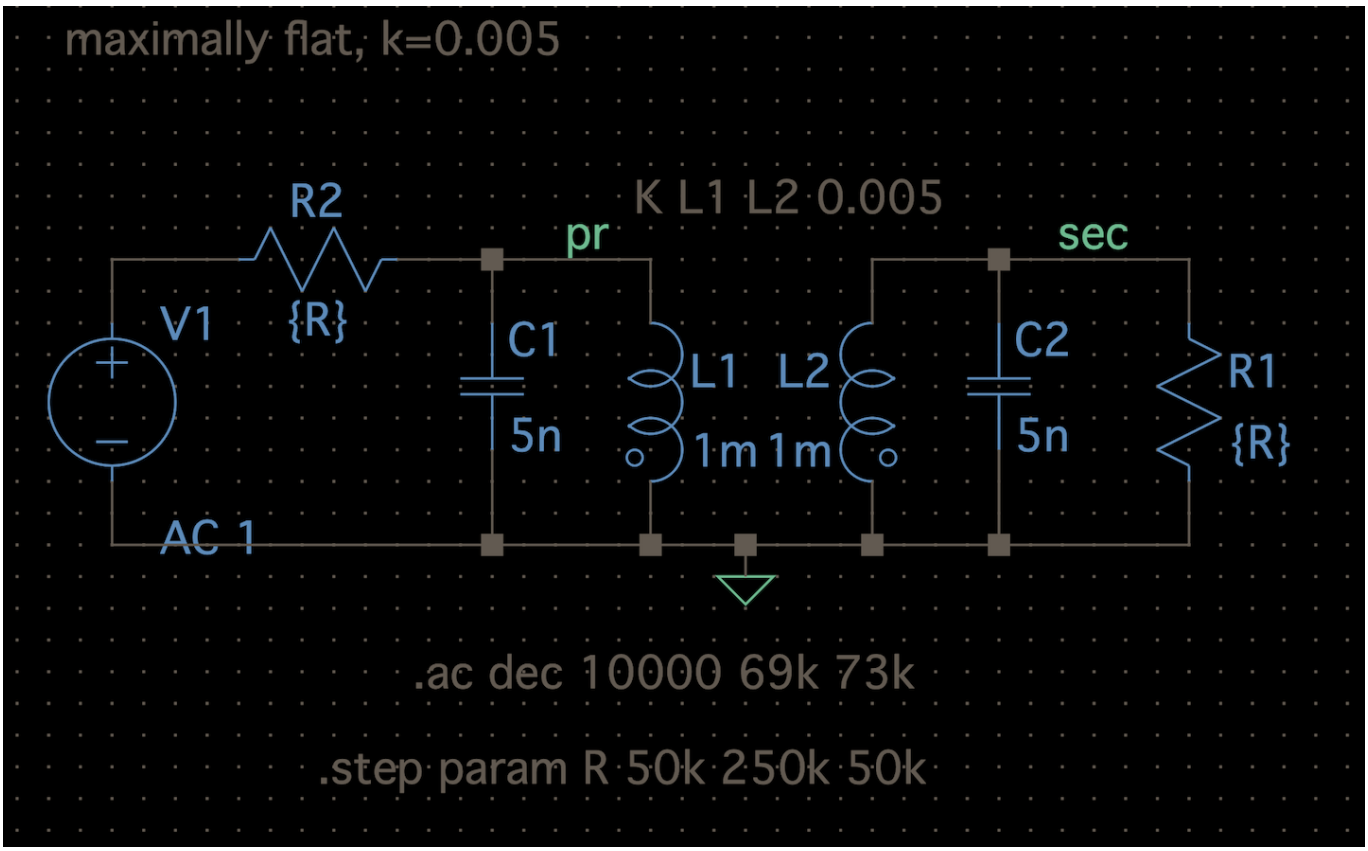
1. Theme: "default"



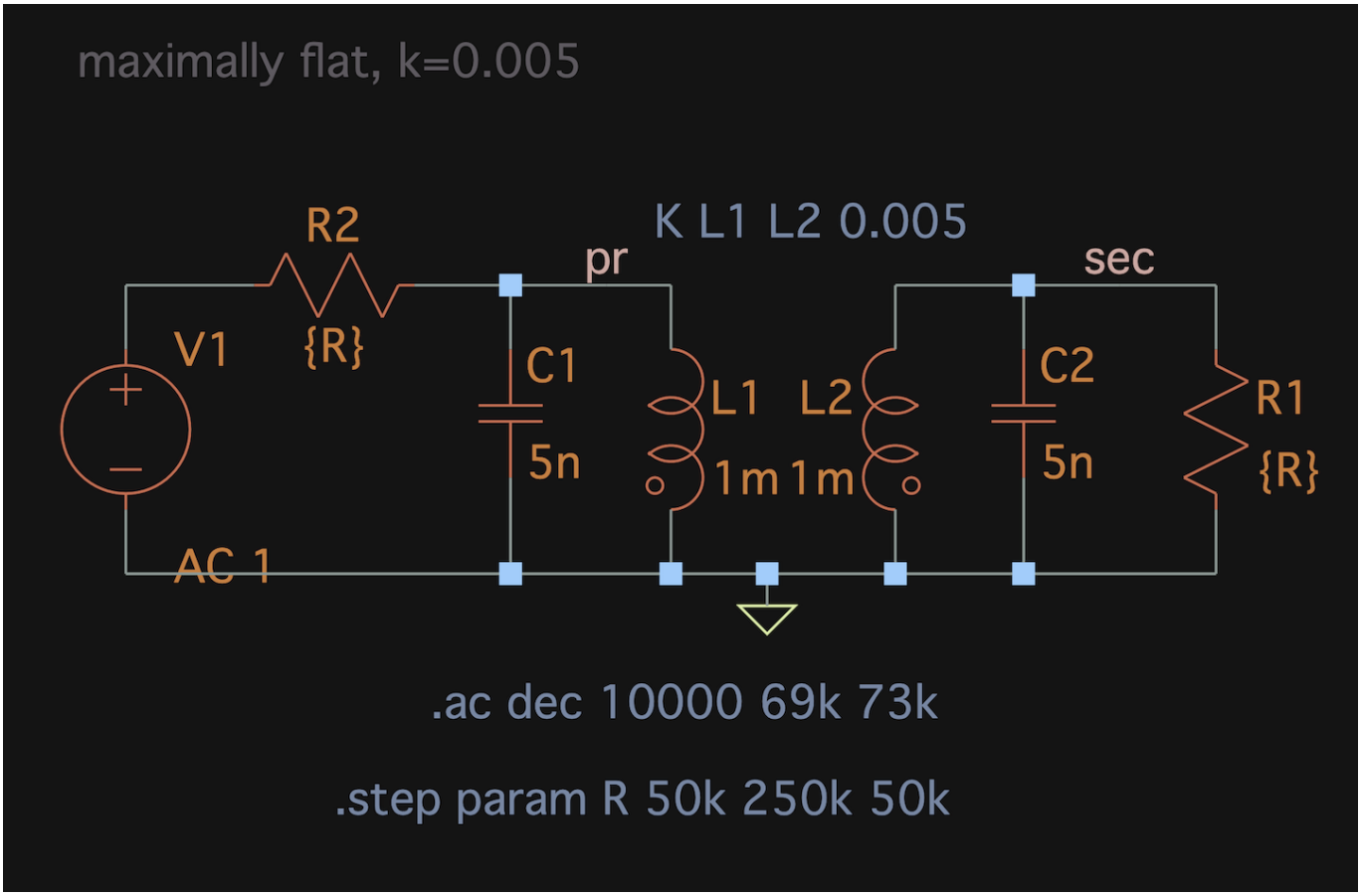
2. Theme: "darcula"



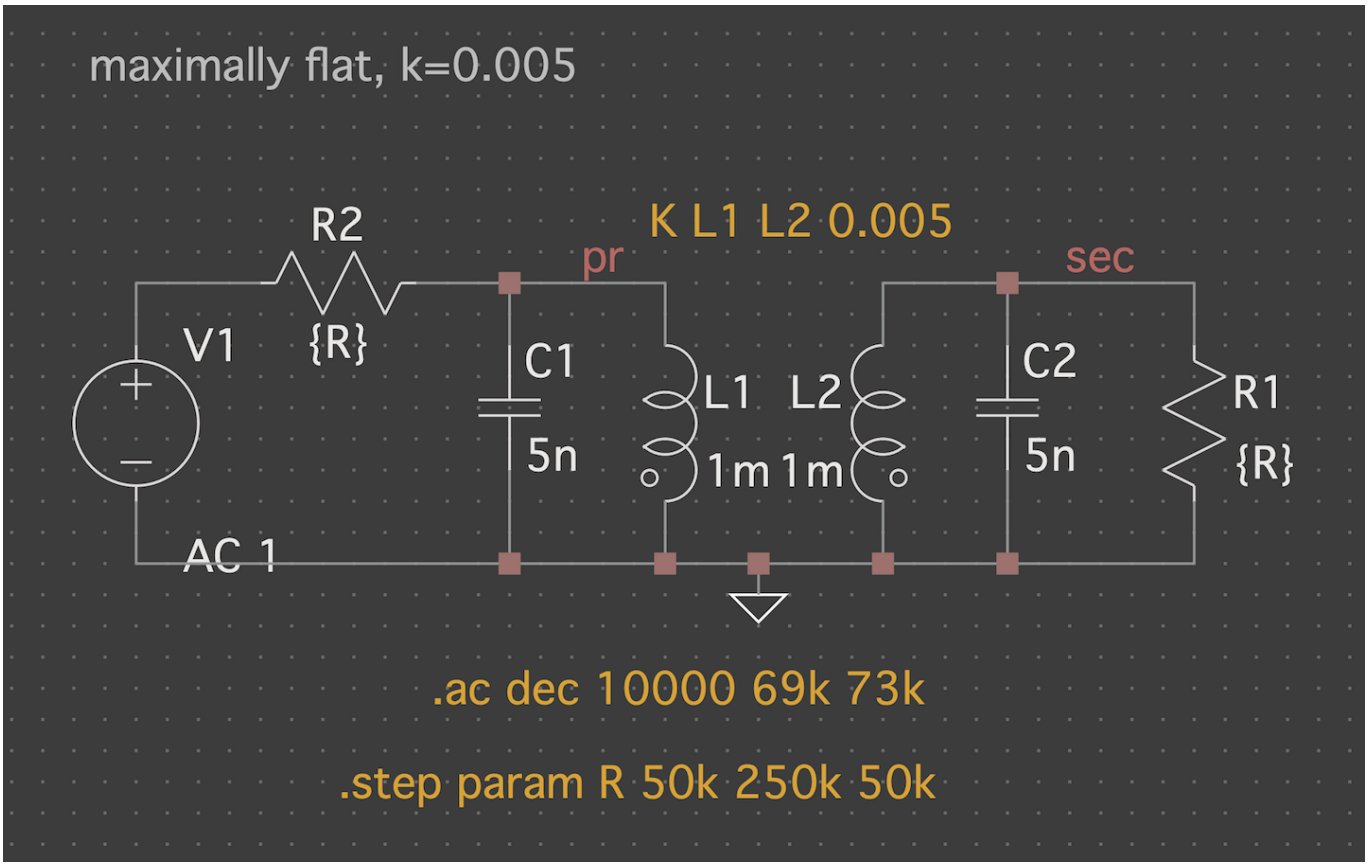
3. Theme: "sakabug"



4. Theme: "twilght-after-dark"



5. Theme: "softdark"



6. Theme: "blackwhite"