Use LaTeX as SLiCAP report generator

Anton J.M. Montagne

May 12, 2025

Contents

Abstract

This document describes how to use LaTeX as SLiCAP report generator and obtain documents with expressions, equations, figures, and tables all updated at document compilation.

1 Introduction

Combining SLiCAP with LaTeX makes it possible to write technical reports while doing the design work. Each time, before before document compilation, figures, tables, graphs, expressions and equations generated by SLiCAP are automatically updated and imported in the LaTeX source.

In this document we will briefly describe the way of working. The LaTeX source code of this document is `tex/SLiCAP_latex.tex`; the path is relative to the SLiCAP project folder. The SLiCAP (Python) source code for this document is `latexReport.py`; in the SLiCAP project folder.

1.1 Project file locations

The project file locations are set and can be altered in the [**projectpaths**] section of the `SLiCAP.ini` file in the SLiCAP project folder. Below the listing of this section for this project:

```
[projectpaths]
html = html/
cir = cir/
lib = lib/
csv = csv/
txt = txt/
img = img/
mathml = mathml/
sphinx = sphinx/
tex = tex/
tex_snippets = tex/SLiCAPdata/
rst_snippets = sphinx/SLiCAPdata/
html_snippets = sphinx/SLiCAPdata/
myst_snippets = sphinx/SLiCAPdata/
md_snippets = sphinx/SLiCAPdata/
```

Below the relevant paths for making SLiCAP LaTeX reports. *All paths are relative to the project folder.*

- `cir`: path to circuit netlist files, e.g. netlist files generated with `makeCircuit()`
- `csv`: path to `.csv` files, e.g. `.csv` files generated with `specs2csv()`
- `tex_snippets`: path to LaTeX snippets generated by the LaTeX formatter.
- `img`: path to image files `.svg, .png, .pdf, gif,` etc., e.g. generated with plot instructions and with `makeCircuit()`.
- `tex`: path to your LaTeX report files and to `preambuleSLiCAP.tex`.

1.2 The preambule

The file `preambuleSLiCAP.tex` imports packages, and defines colors and styles for SLiCAP. It must be imported at the beginning of the document, before **begin{document}**. Below you will find the opening of *this* document:

```
\documentclass[a4paper,12pt]{article}
\input{preambuleSLiCAP.tex}
\title{Use \LaTeX$\,$ as SLiCAP report generator}
```

```
\author{Anton J.M. Montagne}
\begin{document}
```

2 SLiCAP creation of LaTeX output

SLiCAP lets you create LaTeX output in two ways:

- Create images and code listings with SLiCAP functions

  The generation and inclusion of images and code listings in LaTeX will be discussed in section 2.1.

- Use the SLiCAP LaTeX formatter to produce LaTeX snippets

  The generation and inclusion of LaTeX snippets using the formatter will be discussed in section 2.2.

2.1 SLiCAP functions

In the next sections we will describe SLiCAP functions that generate data that can directly be imported by LaTeX.

2.1.1 `makeCircuit()`

if KiCad or Lepton-EDA is used as schematic capture program, `makeCircuit()` creates drawing size images in `.svg` and `.pdf` format in the images folder (see section 1.1 for file locations).

```
# Create a circuit object
cir = sl.makeCircuit("kicad/myPassiveNetwork/myPassiveNetwork.kicad_sch")
```

The LaTeX code to include the schematic circuit diagram of `cir` is:

```
\begin{figure}[h]
\centering
 \includegraphics[width=16cm]{../img/myPassiveNetwork.pdf}
 \caption{Schematic diagram}
 \label{fig-myPassiveNetwork}
\end{figure}
```

The result is shown in Figure 1.

The netlist file that is created with `makeCircuit()` can be displayed in the LaTeX document using:

```
\lstinputlisting[language=ltspice, numbers=left]{../cir/myPassiveNetwork.
    cir}
```

This will render as follows:

```
"myPassiveNetwork"
.source V1
.detector V_out
.param R_s=150 R_ell=50 L=1u C_a=25n C_b=250p S_v=4e-18
.param V_DC = 5 sigma_V=0.05 sigma_R1 = 0.02 sigma_R2=0.01
C1 0 out C value={C_a} vinit=0
C2 out 1 C value={C_b} vinit=0
L1 1 out L value={L} iinit=0
```

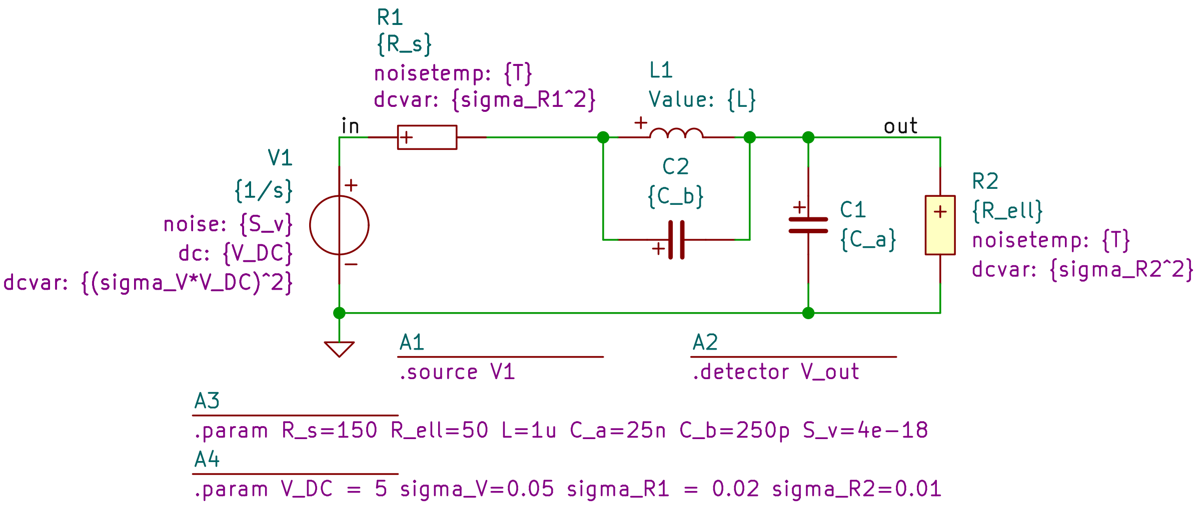

Figure 1: Schematic diagram

```
R1 1 in R value={R_s} noisetemp={T} noiseflow=0 dcvar={sigma_R1^2} dcvarlot=0
R2 out 0 R value={R_ell} noisetemp={T} noiseflow=0 dcvar={sigma_R2^2} dcvarlot=0
V1 in 0 V value={1/s} noise={S_v} dc={V_DC} dcvar={(sigma_V*V_DC)^2}
.end
```

2.1.2 plot(), plotSweep(), and plotPZ()

Figure 2 shows the `dBmag` plot of the source-to-load transfer of the circuit.

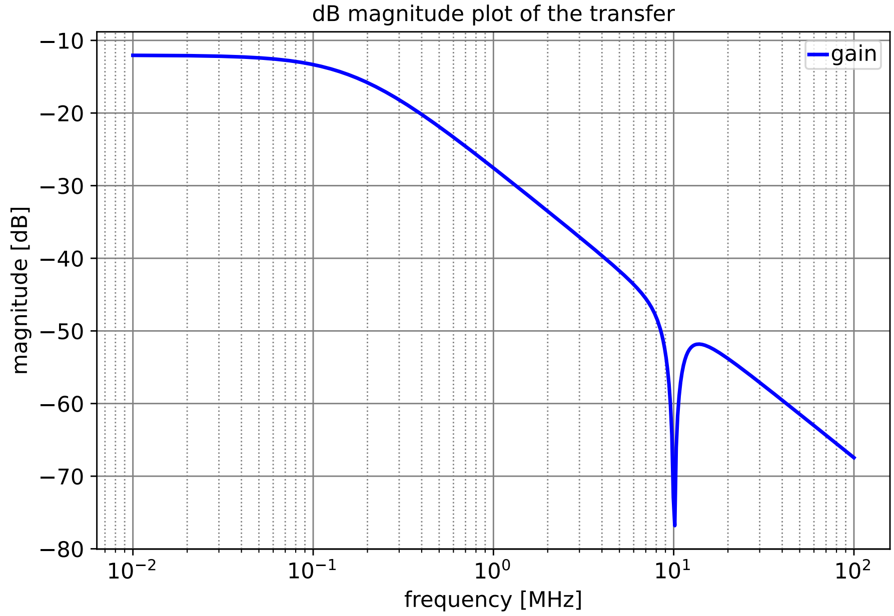

Figure 2: Magnitude plot of the source-to-load transfer

The SLiCAP code for creating this plot is:

```
# Plot the magnitude plot
```

```
result = sl.doLaplace(cir, pardefs="circuit", numeric=True)
sl.plotSweep("dBmag", "dB magnitude plot of the transfer", result, 0.01,
             100, 500, sweepScale="M", funcType="dBmag")
```

The LaTeX source for including it is:

```
\begin{figure}[h]
\centering
 \includegraphics[width=12cm]{../img/dBmag.pdf}
 \caption{Magnitude plot of the source-to-load transfer}
 \label{fig-dBmag}
\end{figure}
```

2.2 LaTeX formatter

The LaTeX formatter in SLiCAP generates LaTeX snippets that can be imported in LaTeX documents using **\input{}** statements. It needs to be initialized. An example of the initialization of this formatter is shown below (see line 12).

```
8  import SLiCAP as sl
9  import sympy as sp
10
11 sl.initProject("LATEX formatter") # Initialize the SLiCAP project
12 ltx = sl.formatter("latex")       # Initialize a LaTeX formatter
```

In the following sections we describe formatter methods (in aplhabetic order) that generate specific LaTeX snippets.

2.2.1 `coeffsTransfer(transferCoeffs, label="", caption="", color="myyellow")`

This method is used to display the numerator and denominator coefficients of a rational function in the form of a table.

Let $H(s) = \frac{R_\ell\left(C_b L s^2+1\right)}{\left(R_\ell+R_s\right)\left(\frac{C_a C_b L R_\ell R_s s^3}{R_\ell+R_s}+\frac{s^2\left(C_a L R_\ell+C_b L R_\ell+C_b L R_s\right)}{R_\ell+R_s}+\frac{s\left(C_a R_\ell R_s+L\right)}{R_\ell+R_s}+1\right)}$.

Table 1 shows the numerator and denominator coefficients of the Laplace variable s of $H(s)$.

Coeff	**Value**
b_0	R_ℓ
b_1	0
b_2	$C_b L R_\ell$
a_0	$R_\ell + R_s$
a_1	$C_a R_\ell R_s + L$
a_2	$L\left(C_a R_\ell + C_b R_\ell + C_b R_s\right)$
a_3	$C_a C_b L R_\ell R_s$

Table 1: Numerator and denominator coefficients of $H(s)$, b_i and a_i, respectively

SLiCAP script:

```
# Coefficients of the transfer:
# Define a transfer function:
H_s = sl.doLaplace(cir).laplace
# Assign the gain, the normalized numerator coefficients and the
```

```
# normalized denominator coefficients to the variable 'coeffs'
coeffs = sl.coeffsTransfer(H_s)
# Generate a LaTeX snippet of the coefficient table with the
# LaTeX formatter 'ltx':
ltx.coeffsTransfer(coeffs, label="tab-coeffs",
                   caption="Numerator and denominator coefficients of " +
                   "$H(s)$, $b_i$ and $a_i$, respectively").save("coeffs")
```

LaTeX code:

```
\input{SLiCAPdata/coeffs.tex}
```

2.2.2 dcvarContribs(resultObject, label="", caption="", color="myyellow")

The individual contributions of independent DC error sources to the detector-referred and source-referred dc variance can be exported in the form of a LaTeX table.

SLiCAP code:

```
dcVarResults = sl.doDCvar(cir)
ltx.dcvarContribs(dcVarResults, label="tab-dcvar",
                  caption="dcvar analysis results").save("dcvar")
```

LaTeX code:

```
\input{SLiCAPdata/dcvar.tex}
```

The result is shown in Table 2.

Name	**Value**	**Units**
V1: Value	$V_{DC}^2\sigma_V^2$	V^2
V1: Source-referred	$V_{DC}^2\sigma_V^2\left(C_bLs^2+1\right)^2$	V^2
V1: Detector-referred	$\frac{R_\ell^2V_{DC}^2\sigma_V^2\left(C_bLs^2+1\right)^2}{\left(R_\ell+R_s\right)^2}$	V^2
I_dcvar_R1: Value	$\frac{V_{DC}^2\sigma_{R1}^2}{\left(R_\ell+R_s\right)^2}$	A^2
I_dcvar_R1: Source-referred	$\frac{R_s^2V_{DC}^2\sigma_{R1}^2}{\left(R_\ell+R_s\right)^2}$	V^2
I_dcvar_R1: Detector-referred	$\frac{R_\ell^2R_s^2V_{DC}^2\sigma_{R1}^2}{\left(R_\ell+R_s\right)^4}$	V^2
I_dcvar_R2: Value	$\frac{V_{DC}^2\sigma_{R2}^2}{\left(R_\ell+R_s\right)^2}$	A^2
I_dcvar_R2: Source-referred	$\frac{R_s^2V_{DC}^2\sigma_{R2}^2}{\left(R_\ell+R_s\right)^2}$	V^2
I_dcvar_R2: Detector-referred	$\frac{R_\ell^2R_s^2V_{DC}^2\sigma_{R2}^2}{\left(R_\ell+R_s\right)^4}$	V^2

Table 2: dcvar analysis results

2.2.3 dictTable(dct, head=None, label="", caption="", color=myyellow")

This method displays the key-value pairs of a dictionary in the form of a table.

SLiCAP code:

```
# Use the dictTable method to display a dictionary as a table
mydct = cir.parDefs
head = ["Name", "Value"]
```

```
52 ltx.dictTable(mydct, head, label='tab-mydct',
53               caption='Circuit parameters using the dictTable format ' +
54               'and modified alternate row color.',
55               color="mygray").save('mydct')
```

LaTeX code:

```
\input{SLiCAPdata/mydct.tex}
```

The result is shown in Table 3. Please notice the change of the default alternate row color as specified in line 52 of the SLiCAP script above.

Name	**Value**
R_s	150
R_ℓ	50
L	$1.0 \cdot 10^{-6}$
C_a	$2.5 \cdot 10^{-8}$
C_b	$2.5 \cdot 10^{-10}$
S_v	$4.0 \cdot 10^{-18}$
V_{DC}	5
σ_V	0.05
σ_{R1}	0.02
σ_{R2}	0.01
T	300

Table 3: Circuit parameters using the dictTable format and modified alternate row color.

2.2.4 `elementData(circuitObject, label="", caption="")`

This method displays the expanded netlist of a circuit object in the form of a table.

SLiCAP code:

```
ltx.elementData(cir, label="tab-expanded",
                caption="Expanded netlist").save("expanded")
```

LaTeX code:

```
```

Table 4 shows the result.

ID	Nodes	Refs	Model	Param	Symbolic	Numeric
C1	0 out		C	value	C_a	$2.5 \cdot 10^{-8}$
				vinit	0	0
C2	out 1		C	value	C_b	$2.5 \cdot 10^{-10}$
				vinit	0	0
L1	1 out		L	value	L	$1.0 \cdot 10^{-6}$
				iinit	0	0
R1	1 in		R	value	R_s	150
				noisetemp	T	300
				noiseflow	0	0
				dcvar	σ_{R1}^2	0.0004
				dcvarlot	0	0
R2	out 0		R	value	R_ℓ	50
				noisetemp	T	300
				noiseflow	0	0
				dcvar	σ_{R2}^2	0.0001
				dcvarlot	0	0
V1	in 0		V	value	$\frac{1}{s}$	$\frac{1}{s}$
				noise	S_v	$4.0 \cdot 10^{-18}$
				dc	V_{DC}	5
				dcvar	$V_{DC}^2 \sigma_V^2$	0.0625

Table 4: Expanded netlist

2.2.5 `eqn(LHS, RHS, units="", label="", multiline=False)`

The formatter method `eqn()` creates a LaTeX snippet of a displayed and numbered equation.

SLiCAP code:

```
# Evaluate the transfer of the network
transfer = sl.doLaplace(cir).laplace

# Save the transfer as a LaTeX displayed equation
ltx.eqn("V_out/V_in", transfer, label="eq-H1").save("H1")
```

LaTeX code:

```
The transfer function is shown in (\ref{eq-H1}).
\input{SLiCAPdata/H1.tex}
```

This renders as:

The transfer function is shown in (1).

$$\frac{V_{out}}{V_{in}} = \frac{R_\ell \left(C_b L s^2 + 1\right)}{\left(R_\ell + R_s\right)\left(\frac{C_a C_b L R_\ell R_s s^3}{R_\ell + R_s} + \frac{s^2 \left(C_a L R_\ell + C_b L R_\ell + C_b L R_s\right)}{R_\ell + R_s} + \frac{s \left(C_a R_\ell R_s + L\right)}{R_\ell + R_s} + 1\right)} \tag{1}$$

If `multiline=True` SLiCAP breaks the equation in parts of a sum or a product.

2.2.6 `eqnInline(LHS, RHS, units="")`

The method `eqnInline()` produces a LaTeX snippet for an inline equation.

SLiCAP code:

```
# Save the transfer as a LaTeX inline equation
ltx.eqnInline("V_out/V_in", transfer).save("H3")
```

LaTeX code:

```
You can write (\ref{eq-H1}) inline as:
\input{SLiCAPdata/H3.tex}.
```

This renders as:

You can write (1) inline as: $\frac{V_{out}}{V_{in}} = \frac{R_\ell\left(C_b L s^2+1\right)}{(R_\ell+R_s)\left(\frac{C_a C_b L R_\ell R_s s^3}{R_\ell+R_s}+\frac{s^2(C_a L R_\ell+C_b L R_\ell+C_b L R_s)}{R_\ell+R_s}+\frac{s(C_a R_\ell R_s+L)}{R_\ell+R_s}+1\right)}$.

2.2.7 expr(expr, units="")

The method `expr()` creates a LaTeX snippet of an inline expression.
SLiCAP code:

```
# Save the transfer as a LaTeX inline expression
ltx.expr(transfer).save("H2")
```

LaTeX code:

```
The transfer can be written as:
\input{SLiCAPdata/H2.tex}.
```

This renders as:

The transfer can be written as: $\frac{R_\ell\left(C_b L s^2+1\right)}{(R_\ell+R_s)\left(\frac{C_a C_b L R_\ell R_s s^3}{R_\ell+R_s}+\frac{s^2(C_a L R_\ell+C_b L R_\ell+C_b L R_s)}{R_\ell+R_s}+\frac{s(C_a R_\ell R_s+L)}{R_\ell+R_s}+1\right)}$.

2.2.8 file(fileName, lineRange=None, firstNumber=None, language=None, style=None)

The method `file()` generates a LaTeX snippet for displaying a code file. The keyword `language` overrides `style`. SLiCAP built-in styles can be seen in `preambule.tex`.
SLiCAP code:

```
f = ltx.file("../cir/myPassiveNetwork.cir", language="ltspice").save("f")
```

Please notice the file path relative to the LaTeX document.
LaTeX code:

```
\input{SLiCAPdata/f.tex}
```

This renders as:
File: myPassiveNetwork.cir

```
"myPassiveNetwork"
.source V1
.detector V_out
.param R_s=150 R_ell=50 L=1u C_a=25n C_b=250p S_v=4e-18
.param V_DC = 5 sigma_V=0.05 sigma_R1 = 0.02 sigma_R2=0.01
C1 0 out C value={C_a} vinit=0
C2 out 1 C value={C_b} vinit=0
L1 1 out L value={L} iinit=0
R1 1 in R value={R_s} noisetemp={T} noiseflow=0 dcvar={sigma_R1^2} dcvar
   lot=0
```

```
R2 out 0 R value={R_ell} noisetemp={T} noiseflow=0 dcvar={sigma_R2^2} dc
    varlot=0
V1 in 0 V value={1/s} noise={S_v} dc={V_DC} dcvar={(sigma_V*V_DC)^2}
.end
```

2.2.9 matrixEqn(Iv, M, Dv, label="")

The method matrixEqn() generates a LaTeX snippet for a displayed matrix equation. Iv, M, and Dv must be Sympy matrices, representing the vector with independent variables, the transfer matrix, and the vector with dependent variables, respectively.

SLiCAP code:

```
# Obtain the MNA matrix equation of this network
matrixResult = sl.doMatrix(cir)
Iv = matrixResult.Iv
Dv = matrixResult.Dv
M  = matrixResult.M

# Save the matrix equation as LaTeX snippet
ltx.matrixEqn(Iv, M, Dv, label="eq-matrices").save("matrices")
```

LaTeX code:

```
The matrix equation of the network is given in (\ref{eq-matrices}).
\input{SLiCAPdata/matrices.tex}
```

This renders as:

The matrix equation of the network is given in (2).

$$\begin{bmatrix} 0 \\ \frac{1}{s} \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -Ls & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & C_b s + \frac{1}{R_s} & -\frac{1}{R_s} & -C_b s \\ 0 & 1 & -\frac{1}{R_s} & \frac{1}{R_s} & 0 \\ -1 & 0 & -C_b s & 0 & C_a s + C_b s + \frac{1}{R_\ell} \end{bmatrix} \cdot \begin{bmatrix} I_{L1} \\ I_{V1} \\ V_1 \\ V_{in} \\ V_{out} \end{bmatrix} \tag{2}$$

2.2.10 netlist(netlistFile, lineRange=None, firstNumber=None)

This method creates an **\input{}** statement for a SLiCAP netlist file.

SLiCAP code:

```
cir = sl.makeCircuit("kicad/myPassiveNetwork/myPassiveNetwork.kicad_sch")

ltx.netlist("myPassiveNetwork.cir").save("netlist")
```

LaTeX code:

```
\input{SLiCAPdata/netlist.tex}
```

This renders as:

Netlist: myPassiveNetwork.cir

```
"myPassiveNetwork"
.source V1
.detector V_out
.param R_s=150 R_ell=50 L=1u C_a=25n C_b=250p S_v=4e-18
.param V_DC = 5 sigma_V=0.05 sigma_R1 = 0.02 sigma_R2=0.01
```

```
C1 0 out C value={C_a} vinit=0
C2 out 1 C value={C_b} vinit=0
L1 1 out L value={L} iinit=0
R1 1 in R value={R_s} noisetemp={T} noiseflow=0 dcvar={sigma_R1^2} dcvar
    lot=0
R2 out 0 R value={R_ell} noisetemp={T} noiseflow=0 dcvar={sigma_R2^2} dc
    varlot=0
V1 in 0 V value={1/s} noise={S_v} dc={V_DC} dcvar={(sigma_V*V_DC)^2}
.end
```

2.2.11 `noiseContribs(resultObject, label="", caption="", color="myyellow")`

The method `noiseContribs()` creates a table with noise sources and their contributions to the source-referred noise and the detector-referred noise.

SLiCAP code:

```
noiseResults = sl.doNoise(cir, pardefs="circuit")
ltx.noiseContribs(noiseResults, label="tab-noise", caption="Noise
    contributions").save("noise")
```

LaTeX code:

```
\input{SLiCAPdata/noise.tex}
```

This renders as shown in Table 5.

Name	Value	Units
V1: Value	$4.0 \cdot 10^{-18}$	$\frac{V^2}{Hz}$
V1: Source-referred	$4.0 \cdot 10^{-18}$	$\frac{V^2}{Hz}$
V1: Detector-referred	$\frac{6.4 \cdot 10^{23} \left(9.87 \cdot 10^{-15} f^{2} - 1\right)^{2}}{8653.0 f^{6} - 1.594 \cdot 10^{18} f^{4} + 8.846 \cdot 10^{31} f^{2} + 2.56 \cdot 10^{42}}$	$\frac{V^2}{Hz}$
I_noise_R1: Value	$1.105 \cdot 10^{-22}$	$\frac{A^2}{Hz}$
I_noise_R1: Source-referred	$2.485 \cdot 10^{-18}$	$\frac{V^2}{Hz}$
I_noise_R1: Detector-referred	$\frac{3.976 \cdot 10^{23} \left(9.87 \cdot 10^{-15} f^{2} - 1\right)^{2}}{8653.0 f^{6} - 1.594 \cdot 10^{18} f^{4} + 8.846 \cdot 10^{31} f^{2} + 2.56 \cdot 10^{42}}$	$\frac{V^2}{Hz}$
I_noise_R2: Value	$3.314 \cdot 10^{-22}$	$\frac{A^2}{Hz}$
I_noise_R2: Source-referred	$\frac{8.284 \cdot 10^{-49} \left(876.7 f^{4} - 1.619 \cdot 10^{17} f^{2} + 9.0 \cdot 10^{30}\right)}{\left(9.87 \cdot 10^{-15} f^{2} - 1\right)^{2}}$	$\frac{V^2}{Hz}$
I_noise_R2: Detector-referred	$\frac{1.325 \cdot 10^{-7} \left(876.7 f^{4} - 1.619 \cdot 10^{17} f^{2} + 9.0 \cdot 10^{30}\right)}{8653.0 f^{6} - 1.594 \cdot 10^{18} f^{4} + 8.846 \cdot 10^{31} f^{2} + 2.56 \cdot 10^{42}}$	$\frac{V^2}{Hz}$

Table 5: Noise contributions

2.2.12 `params(circuitObject, label="", caption="", color="myyellow")`

This method creates a single-column table with names of undefined parameters.

SLiCAP code:

```
ltx.params(cir, label="tab-params",
           caption="Undefined parameters").save("params")
```

LaTeX code:

```
Undefined parameters are given in Table \ref{tab-params}

\input{SLiCAPdata/params}
```

This render as:
Undefined parameters are given in Table **??**
No undefined parameters in: myPassiveNetwork

2.2.13 `parDefs(circuitObject, label="", caption="", color="myyellow")`

This method creates a single-column table with circuit parameter definitions.
SLiCAP code:

```
ltx.parDefs(cir, label="tab-pardefs",
            caption="Circuit parameter definitions").save("pardefs")
```

LaTeX code:

```
Parameter definitions are given in Table \ref{tab-pardefs}

\input{SLiCAPdata/pardefs}
```

This renders as:
Parameter definitions are given in Table 6

Name	Symbolic	Numeric
R_s	150	150
R_ℓ	50	50
L	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$
C_a	$2.5 \cdot 10^{-8}$	$2.5 \cdot 10^{-8}$
C_b	$2.5 \cdot 10^{-10}$	$2.5 \cdot 10^{-10}$
S_v	$4.0 \cdot 10^{-18}$	$4.0 \cdot 10^{-18}$
V_{DC}	5	5
σ_V	0.05	0.05
σ_{R1}	0.02	0.02
σ_{R2}	0.01	0.01
T	300	300

Table 6: Circuit parameter definitions

2.2.14 `pz(resultObject, label="", append2caption="", color="myyellow")`

This method creates LaTeX table snippets for results of pole-zero analysis results.
SLiCAP code:

```
polesResult  = sl.doPoles(cir, pardefs="circuit")
zerosResult  = sl.doZeros(cir, pardefs="circuit")
pzResult     = sl.doPZ(cir, pardefs="circuit")
symZeros     = sl.doZeros(cir)

ltx.pz(polesResult, label="tab-poles", caption="Poles of the transfer").save("poles")
ltx.pz(zerosResult, label="tab-zeros", caption="Zeros of the transfer").save("zeros")
ltx.pz(pzResult, label="tab-pz", caption="Poles and zeros of the transfer").save("pz")
ltx.pz(symZeros, label="tab-symzeros", caption="Symbolic zeros of the transfer").save("symzeros")
```

```python
ltx.expr(pzResult.DCvalue).save("dcValue")
```

LaTeX code for the poles table:

```latex
The numeric poles are listed in Table \ref{tab-poles}.

\input{SLiCAPdata/poles.tex}
```

This renders as:
The numeric poles are listed in Table 7.

#	Re [Hz]	Im [Hz]	f [Hz]	Q
p_1	$-1.701 \cdot 10^5$	0	$1.701 \cdot 10^5$	
p_2	$-2.122 \cdot 10^6$	$9.83 \cdot 10^6$	$1.006 \cdot 10^7$	2.37
p_3	$-2.122 \cdot 10^6$	$-9.83 \cdot 10^6$	$1.006 \cdot 10^7$	2.37

Table 7: Poles of the transfer

LaTeX code for the symbolic zeros table:

```latex
The symbolic zeros are listed in Table \ref{tab-symzeros}.

\input{SLiCAPdata/symzeros.tex}
```

This renders as:
The symbolic zeros are listed in Table 8.

#	f [Hz]
z_1	$-\frac{0.5\left(-\frac{1}{C\_bL}\right)^{0.5}}{\pi}$
z_2	$\frac{0.5\left(-\frac{1}{C\_bL}\right)^{0.5}}{\pi}$

Table 8: Symbolic zeros of the transfer

LaTeX code for the numeric zeros table:

```latex
The numeric zeros are listed in Table \ref{tab-zeros}.

\input{SLiCAPdata/zeros}
```

This renders as:
The numeric zeros are listed in Table 9.

#	Re [Hz]	Im [Hz]	f [Hz]	Q
z_1	0	$1.007 \cdot 10^7$	$1.007 \cdot 10^7$	$\tilde{\infty}$
z_2	0	$-1.007 \cdot 10^7$	$1.007 \cdot 10^7$	$\tilde{\infty}$

Table 9: Zeros of the transfer

LaTeX code for the numeric poles-zeros table:

```latex
The poles and zeros are listed in Table \ref{tab-pz}.

\input{SLiCAPdata/pz.tex}

The DC value of the transfer equals:
\input{SLiCAPdata/dcValue.tex}.
```

This renders as:
The poles and zeros are listed in Table 10.

#	Re [Hz]	Im [Hz]	f [Hz]	Q
p_1	$-1.701 \cdot 10^5$	0	$1.701 \cdot 10^5$	
p_2	$-2.122 \cdot 10^6$	$9.83 \cdot 10^6$	$1.006 \cdot 10^7$	2.37
p_3	$-2.122 \cdot 10^6$	$-9.83 \cdot 10^6$	$1.006 \cdot 10^7$	2.37
z_1	0	$1.007 \cdot 10^7$	$1.007 \cdot 10^7$	$\tilde{\infty}$
z_2	0	$-1.007 \cdot 10^7$	$1.007 \cdot 10^7$	$\tilde{\infty}$

Table 10: Poles and zeros of the transfer

The DC value of the transfer equals: 0.25 .

2.2.15 `specs(specs, specType, label="", caption="", color="myyellow")`

This method exprots a LaTeX snippet for a specification tabel.
SLiCAP code:

```
specs = []
f_min = 10
f_max = 10e6
v_n   = sl.rmsNoise(noiseResults, 'onoise', 10, 1e6)
specs.append(sl.specItem("f_min", "Lower limit noise bandwidth", f_min,
                          units="Hz", specType="performance"))
specs.append(sl.specItem("f_max", "Upper limit noise bandwidth", f_max,
                          units="Hz", specType="performance"))
specs.append(sl.specItem("v_n", "RMS output noise over noise bandwidth",
```

LaTeX code for the performance specifications table:

```
The performance specifications are listed in Table \ref{tab-performance}.

\input{SLiCAPdata/performance.tex}
```

This renders as:
The performance specifications are listed in Table 11.

name	description	value	units
f_{min}	Lower limit noise bandwidth	10	Hz
f_{max}	Upper limit noise bandwidth	$1.0 \cdot 10^7$	Hz

Table 11: Performance specifications

LaTeX code for the design specifications table:

```
The design specifications are listed in Table \ref{tab-design}.

\input{SLiCAPdata/design.tex}
```

This renders as:
The design specifications are listed in Table 12.

name	description	value	units
v_n	RMS output noise over noise bandwidth	$4.558 \cdot 10^{-7}$	V

Table 12: Design specifications

2.2.16 stepArray(stepVars, stepArray, label="", caption="", color="myyellow")

This method exports a LaTeX table snippet with step-data for array-type stepping.
SLiCAP code:

```
sl.specs2csv(specs, "specs.csv")
ltx.specs(specs, specType="performance", label="tab-performance",
          caption="Performance specifications").save("performance")
ltx.specs(specs, specType="design", label="tab-design",
          caption="Design specifications").save("design")

step_dict = {}
step_dict["method"] = "array"
step_dict["params"] = ["C_b", "R_ell"]
```

LaTeX code for the step array table:

```
The step values that apply to Figure \ref{fig-dBmagStepped} are shown in
   Table \ref{tab-stepdict}.

\input{SLiCAPdata/stepdict.tex}
```

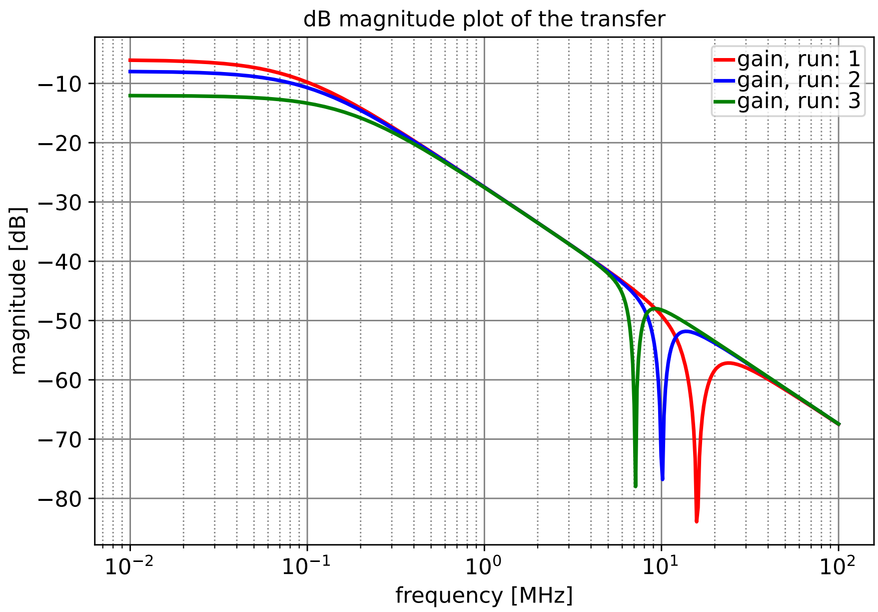

Figure 3: Magnitude plot of the source-to-load transfer

This renders as:
The step values that apply to Figure 3 are shown in Table 13.

	$\mathbf{C_b}$	$\mathbf{R_\ell}$
Run 1:	$1.0 \cdot 10^{-10}$	150
Run 2:	$2.5 \cdot 10^{-10}$	100
Run 3:	$5.0 \cdot 10^{-10}$	50

Table 13: Step array