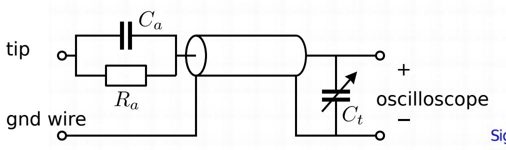
The Oscilloscope Probe

A SLiCAP demonstration of the analysis of linear dynamic circuits

Circuit data

Circuit diagram

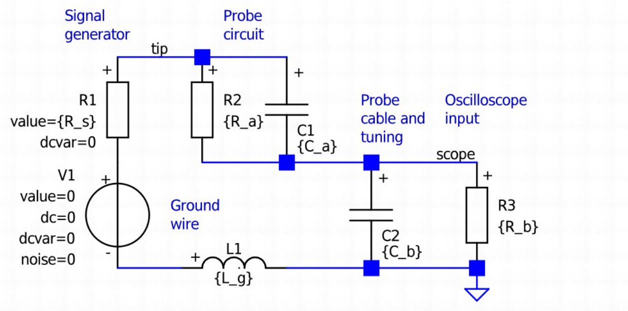
Component model



Voltage attenuation at low frequencies determined by R_a and the input resistance of the oscilloscope.

Voltage attenuation at high frequencies determined by C_a, C_t, the cable capacitance, and the input capacitance of the oscilloscope.

C_t should be tuned for equal attenuation at low and high frequencies.



.param A=10 L_g=10n R_s=50 R_a={(A-1)*R_b} R_b = 1M C_a=15p C_b={C_a*R_a/R_b}

Netlist: ProbeSymbolic.cir

```
"ProbeSymbolic"
* Z:\mnt\DATA\www\SEDwebsite\_build\SLiCAPprojects\ProbeCircuit\cir\ProbeSymbolic.asc
C1 tip scope {C_a}
C2 scope 0 {C_b}
R2 tip scope {R_a}
R3 scope 0 {R_b}
V1 P001 N001 V value=0 dc=0 dcvar=0 noise=0
L1 N001 0 {L_g}
R1 tip P001 r value={R_s} dcvar=0
* Signal \ngenerator
* Probe\ncircuit
* Oscilloscope\ninput
* Probe\ncable and\ntuning
* Ground\nwire
.param A=10 L_g=10n R_s=50 R_a={(A-1)*R_b} R_b = 1M C_a=15p C_b={C_a*R_a/R_b}
.backanno
.end
```

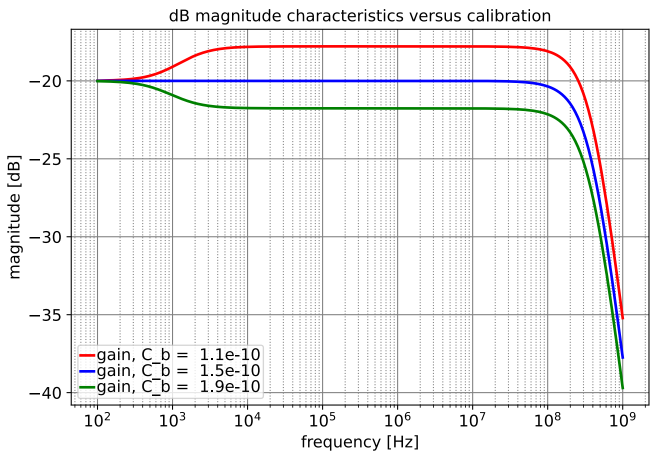
Table: Element data of expanded netlist 'ProbeSymbolic'

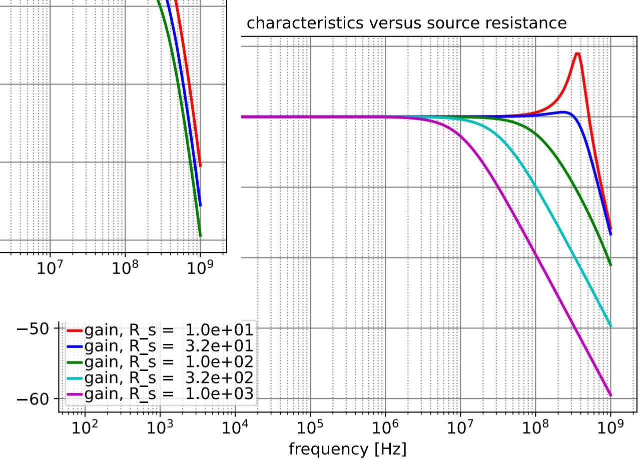
RefDes	Nodes	Refs	Model	Param	Symbolic	Numeric
C1	tip scope		C	value	C_a	$1.5 \cdot 10^{-11}$
C2	scope 0		C	value	C_b	$1.35 \cdot 10^{-10}$
L1	N001 0		L	value	L_g	$1.0 \cdot 10^{-8}$
R1	tip P001		r	value	R_s	50.0
				dcvar	0	0
R2	tip scope		R	value	R_a	$9.0 \cdot 10^{6}$
R3	scope 0		R	value	R_b	$1.0 \cdot 10^{6}$
V1	P001 N001		V	value	0	0
				dc	0	0
				dcvar	0	0
				noise	0	0

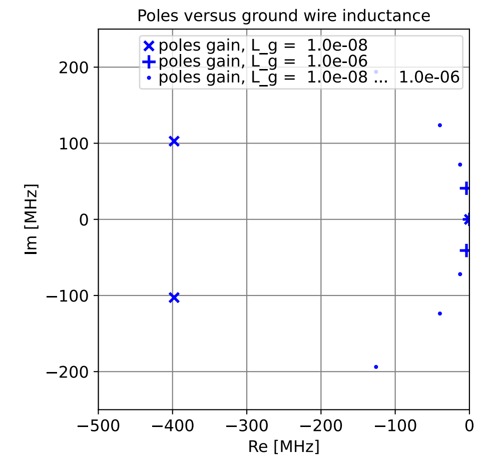
Table: Parameter definitions in 'ProbeSymbolic'.

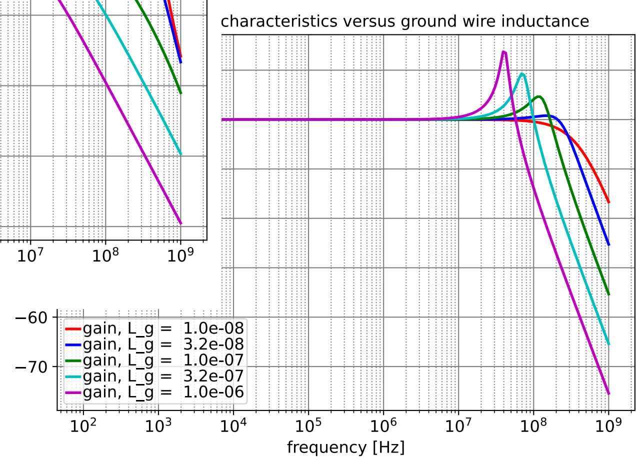
Name	Symbolic	Numeric
A	10	10
C_a	$1.5 \cdot 10^{-11}$	$1.5 \cdot 10^{-11}$
C_b	$\frac{C_a R_a}{R_b}$	$1.35 \cdot 10^{-10}$
L_g	$1.0 \cdot 10^{-8}$	$1.0 \cdot 10^{-8}$
R_a	$R_b (A-1)$	$9.0 \cdot 10^{6}$
R_b	$1.0 \cdot 10^{6}$	$1.0 \cdot 10^{6}$
R_s	50	50.0

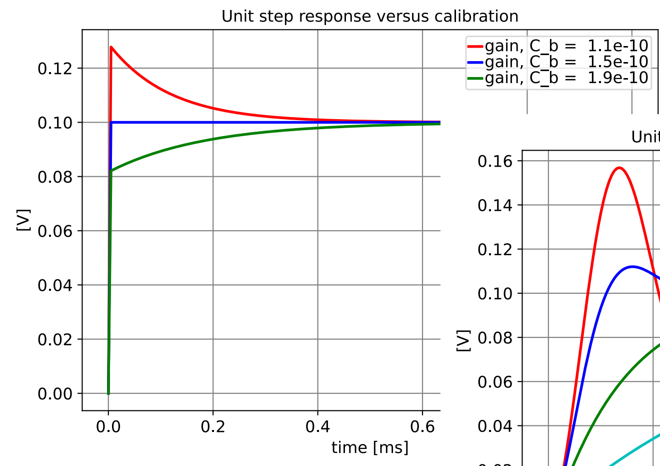
Go to ProbeSymbolic_index

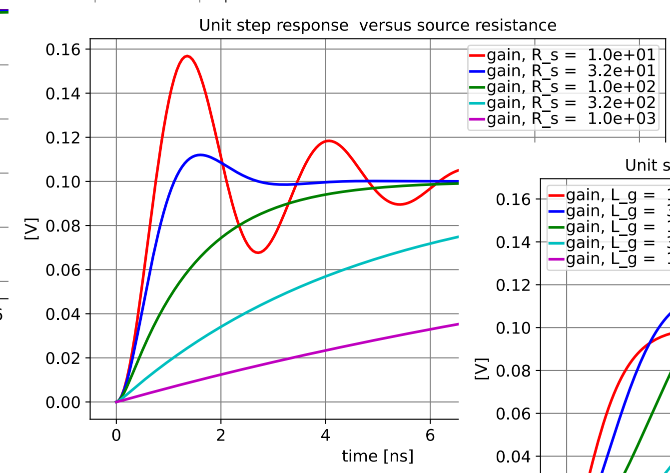


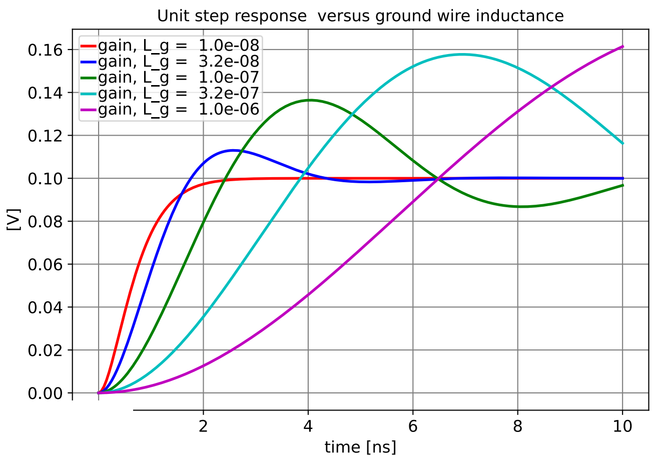












Define variables in a separate file

```
# Values for 10x 15pF probe (350MHz @ 50Ohm)

attn    = 10
R_scoop = 1e6
C_scope = 20e-12

R_src   = 50
L_gnd   = 12e-9
C_cable = 100e-12
C_tune  = 30e-12

R_a     = R_scoop*(attn - 1)
C_a     = (C_cable + C_scope + C_tune)/(attn - 1)
C_b     = C_cable + C_scope + C_tune

SHOW = False
```

Create a project, an instruction, and a circuit

```
from SLiCAP import *
import values

prj = initProject('Probe project')
fileName = 'ProbeNumeric'
#makeNetlist(fileName + '.asc', fileName)
# Create an instance of the instruction object
i1  = instruction()
# Define the circuit for this instruction and check its
i1.setCircuit(fileName + '.cir')
```

Display the circuit data on an HTML page

```
htmlPage("Circuit data")
head2html("Circuit diagram")
img2html('probe.jpg', 100)
img2html(fileName + ".svg", 500)
netlist2html(fileName + ".cir")
elementData2html(i1.circuit)
params2html(i1.circuit)
```

Define the source, the detector, and the gain type

```
i1.setSource('V1')
i1.setDetector('V_scope')
i1.setGainType('gain')
```

Define your own Python Objects and Functions

```
def stepParam(stepParameter):

    if stepParameter == 'cal':
        # Define and enable stepping of calibration +/-25%
        i1.setStepVar('C_b')
        i1.setStepStart(0.75*C_b_comp)
        i1.setStepStop(1.25*C_b_comp)
        i1.setStepNum(3)
        i1.setStepMethod('lin')
        i1.stepOn()

    elif stepParameter == 'gnd':
        # Define and enable stepping of ground wire inductance
        i1.setStepVar('L_g')
        i1.setStepStart(10e-9)
        i1.setStepStop(1e-6)
        i1.setStepNum(5)
        i1.setStepMethod('log')
        i1.stepOn()

    elif stepParameter == 'src':
        # Define and enable stepping of source resistance
        i1.setStepVar('R_s')
        i1.setStepStart(10)
        i1.setStepStop(1e3)
        i1.setStepNum(5)
        i1.setStepMethod('log')
        i1.stepOn()
    else:
        i1.stepOff()

stepDict = {'cal' : 'calibration', 'gnd': 'ground wire inductance',
            'src': 'source resistance'}

stepParams = list(stepDict.keys())
stepParams.sort()
```

Create plots and place them in your HTML report

```
###########################################################################
#                                                                         #
# Frequency response versus detuningg, wire inductance, and source resistance #
#                                                                         #
###########################################################################

htmlPage('Frequency response')
head2html("Circuit diagram")
img2html('probe.jpg', 100)
img2html(fileName + ".svg", 500)

i1.setDataType('laplace')

for par in stepParams:

    head2html("Frequency response versus " + stepDict[par])
    stepParam(par)
    result = i1.execute()

    plotMag = plotSweep('dBmagProbe_' + par, 'dB magnitude characteristics ' +
                        'versus ' + stepDict[par],
                        result, 100, 1e9, 200, funcType='dBmag',
                        show=values.SHOW)

    plotPhs = plotSweep('phaseProbe_' + par, 'Phase characteristics '
                        'versus ' + stepDict[par],result,
                        100, 1e9, 200, funcType='phase', show=values.SHOW)

    plotDly = plotSweep('delayProbe_' + par, 'Delay characteristics '
                        'versus ' +stepDict[par], result,
                        100, 1e9, 200, funcType='delay', show=values.SHOW)

    plotDlyZoom = plotSweep('delayProbeZoom' + par, 'Delay characteristics ' +
                            'versus ' +stepDict[par], result, 1e6, 1e9, 200,
                            funcType='delay', show=values.SHOW)

    fig2html(plotMag, 600)
    fig2html(plotPhs, 600)
    fig2html(plotDly, 600)
    fig2html(plotDlyZoom, 600)
```

Perform symbolic analysis and create HTML reports

```
###########################################################################
#                                                                         #
#                       Symbolic matrix equation                          #
#                                                                         #
###########################################################################

# Create an HTML page to display the results

htmlPage("MNA matrix equation")
head2html("Circuit diagram")
img2html('probe.jpg', 100)
img2html(fileName + ".svg", 500)

# Put some text on the HTML page
text2html('The MNA matrix equation of the circuit is:')

i1.setGainType('vi')
i1.setDataType('matrix')
result = i1.execute()
matrices2html(result)

###########################################################################
#                                                                         #
#         Symbolic analysis of the Laplace transfer of the gain           #
#                                                                         #
###########################################################################

# Create an HTML page to display the results

htmlPage("Symbolic Laplace Transform")
head2html("Circuit diagram")
img2html('probe.jpg', 100)
img2html(fileName + ".svg", 500)

# Define the source and the detector, and the gain type.

i1.setSource('V1')
i1.setDetector('V_scope')
i1.setGainType('gain')

# Laplace transform of the transfer

i1.setSimType('symbolic')
i1.setDataType("laplace")
result = i1.execute()

symbolic_laplace = result.laplace

text2html("The Laplace Transform of the transfer $\\frac{V_scope}{V_s}$ is:")

eqn2html("V_scope/V_s", symbolic_laplace)

# Let's see the expression for in case the probe is calibrated

text2html("If the probe is correctly calibrated, we substitute:")
eqn2html(C_b, C_a*R_a/R_b)

text2html("The expression of the transfer then simplifies to:")
# Use sympy substitution
symbolic_laplace_calibrated = symbolic_laplace.subs(C_b, C_b_comp)
eqn2html("V_scope/V_s", symbolic_laplace_calibrated)

text2html("If we factorize this result we obtain:")
# Use sympy factorization
symbolic_laplace_calibrated = sp.factor(symbolic_laplace_calibrated)
eqn2html("V_scope/V_s", symbolic_laplace_calibrated)

text2html("If we normalize this result, we obtain:")
# Use the SLiCAP "normalizeRational" function to see the DC transfer and the
# frequency-dependent part
symbolic_laplace_calibrated = normalizeRational(symbolic_laplace_calibrated)
eqn2html("V_scope/V_s", normalizeRational(symbolic_laplace_calibrated))
```

The 1-rst order ($R_s = 0$) detuned unit step response is found as:

$$\mu_t = \frac{R_b}{R_a + R_b} - \frac{\left(-C_a R_a^2 R_b + C_b R_a R_b^2\right) e^{-\frac{t(R_a+R_b)}{R_a R_b (C_a+C_b)}}}{R_a R_b \left(C_a + C_b\right)\left(R_a + R_b\right)} \tag{1}$$

The 1-rst order ($R_s \neq 0,\ C_b = C_a \frac{R_a}{R_b}$) tuned unit step response is found as:

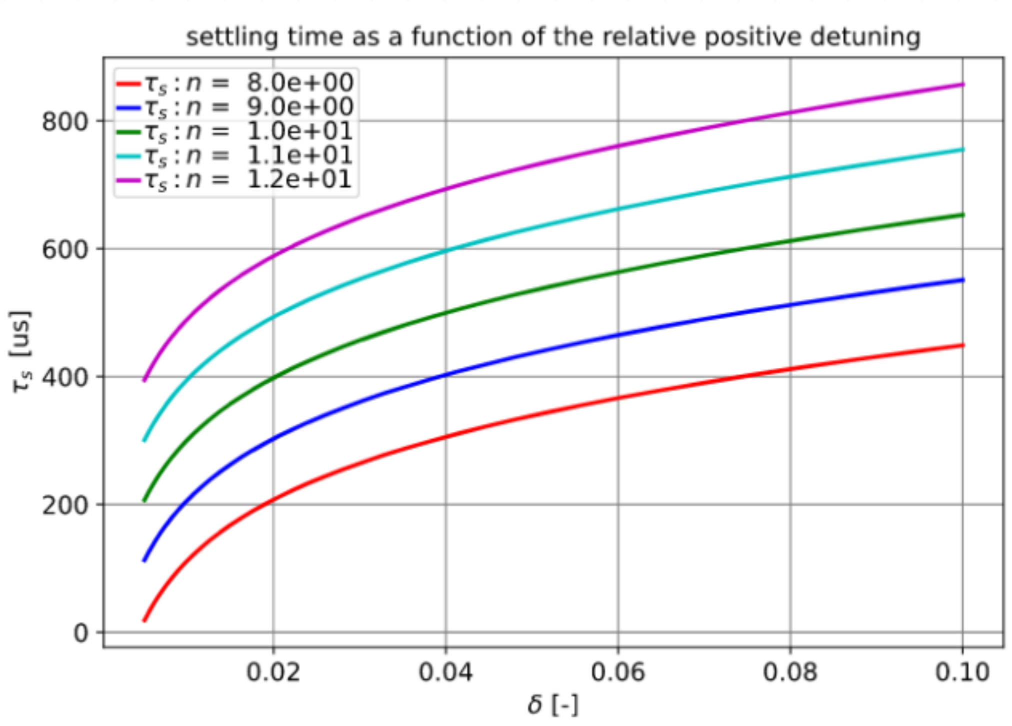
$$\mu_t = \frac{R_b}{R_a + R_b + R_s} - \frac{R_b e^{-\frac{t(R_a+R_b+R_s)}{C_a R_a R_s}}}{R_a + R_b + R_s} \tag{2}$$

The settling error to the final value of a wrongly tuned probe can be written as a function of time:

$$\epsilon_t = \frac{(C_a R_a - C_b R_b) e^{-\frac{t(R_a+R_b)}{R_a R_b (C_a+C_b)}}}{\left(C_a + C_b\right)\left(R_a + R_b\right)} \tag{3}$$

After equating this value with a 1 LSB error, we find the settling time as function of the relative positivedetuning δ of C_b:

$$\tau_s = \frac{9 C_a R_b \left(9\delta + 10\right) \log\left(\frac{9 \cdot 2^n \delta}{9\delta + 10}\right)}{10} \tag{4}$$



What is SLiCAP

- SLiCAP is an acronym for: **S** ymbolic **Li** near **C** ircuit **A** nalysis **P** rogram
- SliCAP is a tool for **algorithm-based analog design automation**
- SLiCAP is intended for setting up and solving **design equations** of electronic circuits
- SLiCAP is a an **open source** application written in Python and maxima CAS
- SLiCAP is part of the tool set for teaching 'Structured Electronic Design' at the Delft University of Technology

Why should you use SLiCAP

- SLiCAP facilitates analog design automation
- SLiCAP speeds up the circuit engineering process
- SLiCAP makes complex symbolic math doable
- SLiCAP integrates documentation and design
- SLiCAP facilitates design education and knowledge building

Features

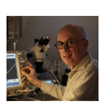
- Accepts SPICE-like netlists as input
- Concurrent design and documentation
- Supports and facilitates structured analog design

Capabilities

- Conversion of hierarchically structured SPICE netlist into mixed symbolic/numeric matrix equation
- Symbolic and numeric noise analysis
- Symbolic and numeric noise integration over frequency
- Symbolic and numeric determination of transfer functions and polynomial coefficients of transfer functions
- Symbolic and numeric determination of the Routh array
- Symbolic and numeric inverse Laplace Transform
- Symbolic and numeric determination of network solutions
- Accurate numeric pole-zero analysis
- Root-locus analysis with a arbitrarily selected circuit parameters as root locus variable(s)
- Symbolic and numeric DC and DC variance analysis for determination of budgets for resistor tolerances and offset and bias quantities
- Symbolic and numeric derivation and solution of design equations for bandwidh, frequency response, noise performance, dc variance and temperature stability

Technology

- Python, Maxima CAS, HTML, CSS, LaTeX, MathJax, Python, Jupyther Lab



(c) 2022 Anton Montagne
anton@montagne.nl
https://analog-electronics.tudelft.nl