Circuit Solver

Informative session

Saurabh Sant, Dr. sc. ETH

saurabh64sant@gmail.com

SemiVi LLC
Zollikon, Switzerland.

May 12, 2025

Overview

Table of Contents

Salient features

The circuit solver takes circuit netlists in spice format and performs desired set of simulations.

Salient features –

- Supports non-linear circuit solver,
- Parameterized electronic components (e.g. diode, BJT, MOSFET),
- Enables user-defined parameters and math functions,
- Python scripting enabled –
  - Component parameters can be modified and results can be read in python file,
  - Data-processing and optimization libraries in python can be readily used.
- User-defined functional models can be created in Python.
- Coupled electro-thermal solver enabled in DC, AC, and transient analyses.

Table of Contents

Supported circuit components

The circuit solver supports the following components –

- Linear components – R, L, C, and mutual-inductance,
- Voltage- and current-controlled switches,
- Lossless and lossy transmission lines, RC network,
- Non-linear components – *Diode, JFET, MOSFET, MESFET*,and *BJT*,
- AC, DC, and transient I/V sources,
- *Dependent sources* – VCVS, CCVS, VCCS, CCCS.
- *Subcircuits* – can be parametrized and math functions can be defined.
- *Heat generation* – enabled in R, diode, JFET, MOSFET, MESFET, and BJT,
- *Heat conduction*

We are open to collaborating with the users to add customized non-linear components.

Supported circuit analyses

The circuit solver supports the following analyses –

- DC ramp up of V/I sources
  - Employs non-linear solver based on damped-Newton's method
- AC analysis
  - Small-signal models of non-linear components
  - Desired bias point achieved by DC ramp
- Transient analysis – time evolution of the given system
  - Backward Euler or Trapezoidal time-stepping methods
- Electro-thermal analysis – Coupled solver for electrical and thermal network
  - Coupling of currents and voltages $\leftrightarrow$ heat flow enabled
  - Coupled electro-thermal solver is enabled in all the DC, AC, and Transient analyses

Contact us if you need to do a specific analysis which cannot be performed with the existing features.

Parameterized electronic components

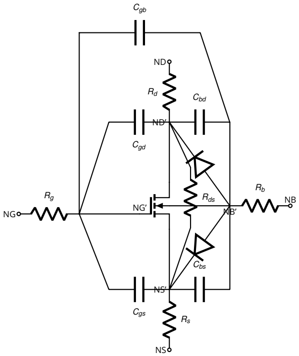


Figure: MOSFET model

Table: MOSFET Parameters

Parameter	Description	Default	Unit
L	channel length	1.	m
W	channel width	1.	m
AS	Source diffusion area	0.	m^2
AD	Drain diffusion area	0.	m^2
VTO	zero-bias threshold voltage	1.	V/K
KP	trans-conductance coefficient	0	$A/V^2/m^2$
...	...	...	...
...	...	...	...

```
.model nmos_depl NMOS (KP=200u VTO=0.6 PHI=0.6 GAMMA=0 LAMBDA=1E-4
+ RDS=1E3 CGSO=1E-10 CGDO=1E-10 CGBO=1E-10 IS=1E-14)
...
M1 Vout Vg Vs Vs nmos_depl
...
```

Parameterized components + good documentation = Easy to create new libraries.

Contact us if you wish to add more parameters.

User-defined Parametrization

```
.subckt Example_1   5   12   18 PARAMS: res1=2.0 res2=12.0
.PARAM res3={res2*2}
.PARAM res4={res3*5}
Rk     5    12   res1
Rl     18   15   res4
Vprob 15     12   DC 0
Rm     18     5   {2.0+I(Vprob)}
.ends

Vs     1     0    DC    1. AC 1. 0 SIN(0V 2.V 1.)
Ra     1     2    1.0
X1     2     0     0    Example_1 PARAMS: res1=5.0 res2=0.01
```

- Defining parameters and functions[1].
- Use Parametrization in main circuit and sub-circuits – increased reuse-ability.
- Import external spice sub-circuit libraries.

User-defined parametrization broadens the possibilities of reuse of the libraries.

[1] Parameters and functions are parsed using 'exprtk' parser.

Table of Contents

Python interface

Python-interface enables the user to perform the following tasks using python scripts.

- Reading a circuit from spice netlist file.

```
import circuitsolver as cs
import numpy as np
p = cs.circuit ()
p.readSpiceCircuitFile ("CircuitTrial.cir")
```

- Reading DC, AC, or transient solutions in python

```
p.getDCSolutionNumpyArray (analysisId=dc1)
p.getNodeTransCurrentNumpyArray(Node="X1 Vs", Component="R3", analysisId=dc1)
```

- Modifying component parameters in python script.

```
R3Now = p.getComponentParamVal (Component="X1 R3")
p.setComponentParamVal (Component="X1 R3", Value=1E2)
```

- Using python scripts for calibration and/or optimization.
    - Optimization libraries in python can be used.

Python interfacing enables usage of python data-processing and optimization libraries while calling the circuit solver.

Python-based functional models

Functional model: A quick and simple way of modeling functionality of a digital chip (for ex. gate-drivers).

- Define a python class and implement driver logic in `updateOutputPinVoltages` function.

```
class SimpleDriver (cs.functionalmodel):
  def updateOutputPinVoltages (self, isOutputPin, inputV, time):
    outV = []
    for i in range(numPins):
      if isOutputPin[i]:
        if time > 1.0: # transient voltage ramp
          outV.append (1.0)
  return outV
```

- Create a circuit spice netlist with a generic N-pin instance (`W1`). List output pins (`OUTPINS`).

```
W1    1    0    DriChp OUTPINS=(1)
Rm    1    out  1000
Cn    out  0    1E-3
.end
```

- Link a new instance `driv1` of user-defined driver class `SimpleDriver` to the N-pin instance (`W1`).

```
driv1 = SimpleDriver ()
p.setFunctionalModel ("W1", driv1)
p.solve ()
```

Functional modeling in python allows users to define python data processing functions (e.g. Fourier transforms) and verify the user-proposed logic in a realistic circuit simulation.

Python-based behavioural models

Behavioural model: A custom compact model of non-linear devices defined in python.

- Create custom diode model by inheriting the `behaviouralmodel` class.
- Calculate pin-currents from pin-voltages in `getPinCurrents`.
- Nonlinear devices: Calculate currents and $\frac{dI_i}{dV_j}$ in `getDerivativesAndPinCurrents`. Auto-differentiation libraries can be used.
- Create a circuit spice netlist with a generic instance (`WB0`). List parameters.

```
R0 2 3 1000
V0 3 0 DC 10
WB0 2 0 MyDiode IS=0.0001
.end
```

- Link a new instance of user-defined driver class `MyDiode` to the instance (`WB0`).

```
dio1 = MyDiode ()
p.readSpiceCircuitFile ("RCckt.cir")
p.setBehaviouralModel ("WB0", dio1)
p.solve ()
```

```
import circuitsolver as cs
import autograd as ad
from autograd.variable import Variable

class MyDiode (cs.behaviouralmodel):
  def __init__(self):
    cs.behaviouralmodel.__init__(self)
    self.Is = 1E-6 # member variables

  def isNonlinear (self):
    return True;

  def useNumericDifferentiation (self):
    return False;

  def setParameter (self, name, value):
    if (name == "IS"):
      self.Is = value

  def getDerivativesAndPinCurrents (self, inputV, time):
    bigV = Variable (inputV)
    va, vc = bigV[0], bigV[1]

    # Anode current
    Ia = self.Is * ad.exp( (va - vc) / self.VT / self.N)
                  + 1E-12 * (va - vc)
    # cathode current
    Ic = - Ia

    # computing gradients
    Ia.compute_gradients()
    Ic.compute_gradients()
    outI = np.append(Ia.data, Ic.data)
    outdI = np.append(Ia.gradient, Ic.gradient)
    out = np.append (outdI, outI)
    return out;
```

Optimizer is useful for *compact model calibration* and *circuit performance optimization*.

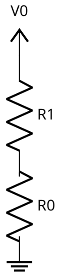


Figure: Resistive divider.

- Define the circuit to be optimized and the analysis.

```
.DC V0 0 10 1
R0 2 0 10
R1 2 1 100
V0 1 0 DC 10 AC 10 1
.end
```

- Set the optimizer algorithm, max iterations, stopping criteria.
- Add parameter to be optimized / fitted e.g. value of `R0`.
- Add target to the optimizer, e.g. power in `R0`.
- Power in `R0` to be maximized → use '-1' as scaling.
- For calibration, measurements are supplied in a csv file whose name is set with `FileName`.
- Custom expressions can be used for optimization.

```
import circuitsolver as cs

# define circuit
p = cs.circuit()
p.readSpiceCircuitFile ("CircuitTrial.cir")

# define optimizer
f = cs.optimizer(p)

f.setOptimizationAlgorithm(Algorithm = "NM-Simplex",
      MaxIter=100,GradientErrorTol=1E-10,
      RelativeErrorTol=1E-5)

f.addOptimizationParameter(Component="R0",
      Param="Value", StartValue=10,
      MinValue=1, MaxValue=200)

f.addExperimentalDataAndPower(AnalysisType="DC",
      AnalysisId=0, FileName="", VNodePlus="2",
      VNodeMinus="0", INode="2", ComponentName="R0",
      Scaling=-1, IntegrationStart=0,
      IntegrationEnd=10, MinValue=-10000,
      MaxValue=10000, PenaltyFactor=10)

# perform optimization
f.optimize()
# read output
out = f.getOptimizedValues()
```

Table of Contents

Detailed user-manual and examples

The user-manual describes in detail the following –

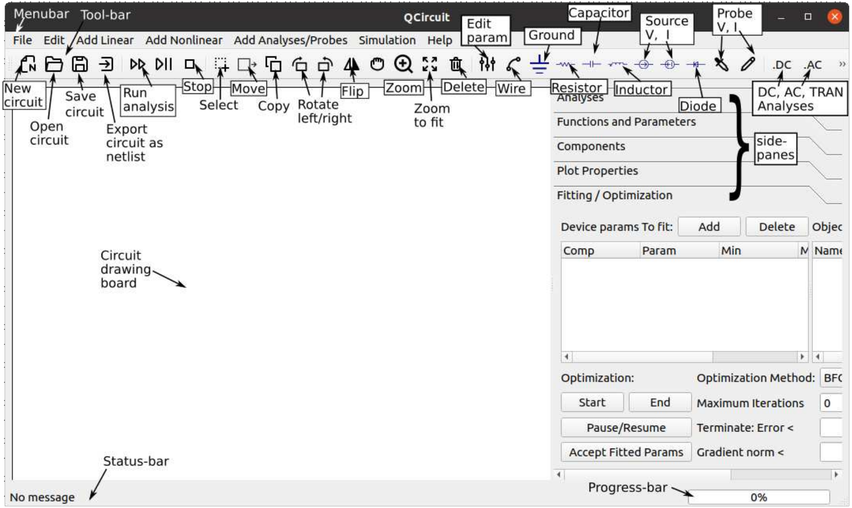
- Equations corresponding to each of the components,
- Parameters of various components and their usage,
- Various analyses and numeric parameters suitable for them,
- Python interface and various functions therein.

Example circuit files and python scripts provided with the distribution can act as quick-references and a starting point for your case.

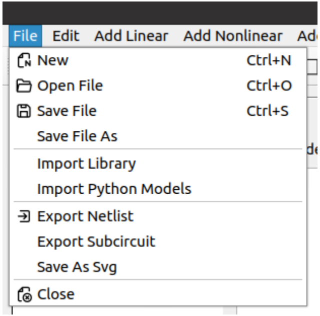
A thorough and clearly-written user-manual enables rapid development of your code.

Table of Contents

GUI - Main window



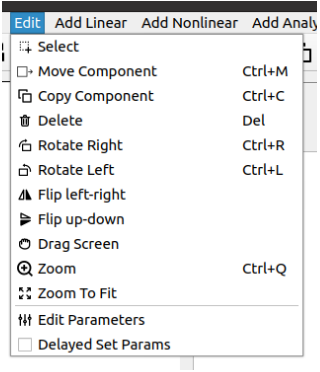
(a) Main window



(b) File menu

- `New` Open new `circuitdraw` window.
- `Open File` Selected '*.cktdr' file opened in a new window.
- `Save File (As)` Current (New) '*.cktdr' file is stored.
- `Import Library` Load all the models/subcircuits in selected *.lib file.
- `Import Python Models` Load functional or behavioural models in '*.py' file.
- `Export Netlist` Spice netlist of the circuit, libraries, analyses, and user-defined functions/parameters stored in spice format.
- `Export Subcircuit` Spice netlist of the circuit, and user-defined functions/parameters are stored in the file as a *sub-circuit*.
- `Save As Svg` Circuit diagram stored in SVG format.

GUI - Edit Menu

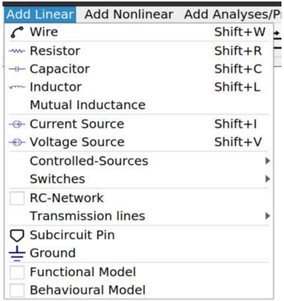


(c) Edit menu

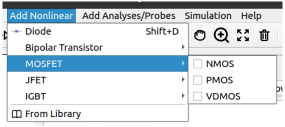
- `Select`When active, draw any arbitrary rectangle in the circuit drawing window to select. The selected components and wires are painted in *red*.
- `Move`When active, press left mouse button on the component, drag the cursor to the desired location, and relase it to move there.
- `Copy` When active, press left mouse button on a component, drag the cursor to the desired location, and relase it to paste.
- `Delete`When active, left mouse click on component or wire will delete it.
- `Rotate` When active, left mouse click on component will rotate it.
- `Flip` When active, left mouse click on component will flip it.
- `Drag Screen` When clicked, ‘Drag-screen’ action is toggled. When ‘Drag-screen’ is active, drag the circuit drawing screen by press → drag → release left mouse button anywhere on the screen.
- `Zoom` Circuit is zoomed to the rectangle in the circuit drawing window
- `Zoom to fit` The circuit diagram is zoomed to fit the window
- `Edit Parameters` When active, left mouse click on any of the component will open the component properties window for that component.
- `Delayed Set Params`

Various actions can be ‘toggled’. This means, if the given action is active, clicking on the action will deactivate it.

GUI - Add Component



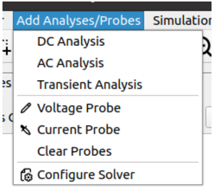
(d) Add linear component



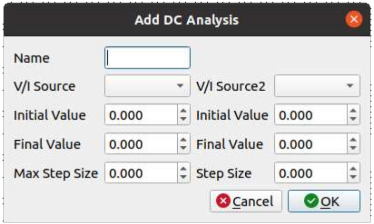
(e) Add non-linear component

Various components are added to the drawing board as follows.

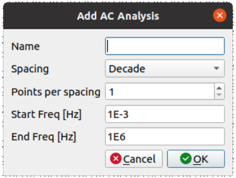
- `Wire` : Click on wire icon and click on or near any component pin. A wire starting at the pin will be drawn. Click on another pin or to end the wire.
- `Component` : Click on the component to activate it. Click on the drawing board to place the component.
- `Mutual Inductance` : Select any two more inductors (L) on the board and then click on `Mutual Inductance` to create a mutual inductance between the selected inductors.
- `Functional/Behavioural model` : Click on the respective icon. Select the model to be placed. Then click on the board to place the model icon.
- `From Library` : Click on the icon to open the dialog box listing all the available models of components. Filter them as per your requirement and select the component to place it.



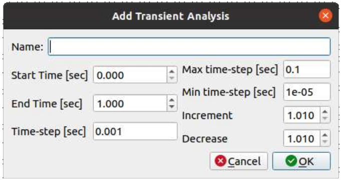
(f) Add Analyses Menu



(g) DC Analysis dialog box



(h) AC Analysis dialog box



(i) Transient Analysis dialog box

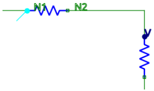
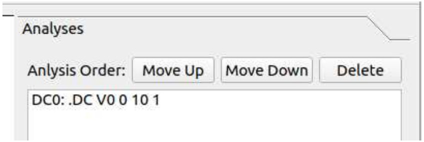


Figure: I/V Probes

Convenient way to add analyses, probes to the circuit –

- `Add Analysis` : Click desired analysis and a dialog box opens. Enter all the analysis parameters and press OK.
- `Add V/I_probe` : Activate V/I-probe and click on the component pin to place the probe. Remember that currents and voltages of the pins can be plotted only if the probes are placed.
- `Clear Probes` : Clear all the probs added to the circuit board.
- `Configure Solver` : Select solvers. set parameters for iterative solvers.

GUI-Side-panes - analyses and parameters



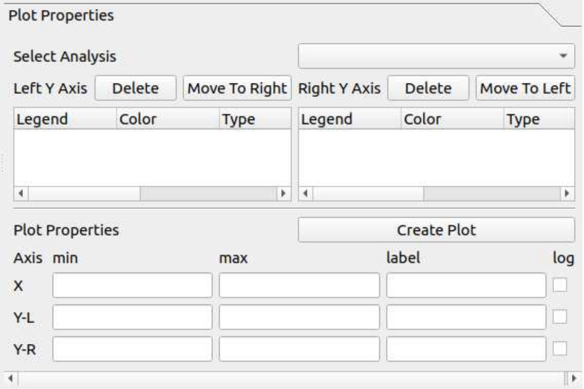
(a) 'Analyses' tab in the side-pane

(b) 'Functions and parameters' tab in the side-pane.

- Analyses are executed in the same order as the appear here.
- Move Up / Move Down : Move selected analysis up/down.
- Delete : Delete selected analysis.
- `Double Click` analysis to edit it.

- New Parameter : Add new parameter (`.PARAM`).
- New Function : Add a new function (`.FUNC`).
- Validate : Check if any parameter used in the function is not defined yet.
- Delete : Delete selected parameter.

GUI-Side-panes - Plot Curves

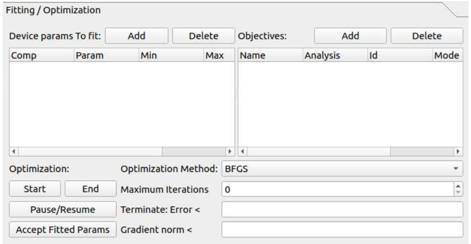


(c) ‘Plot Properties’ tab in the side-pane.

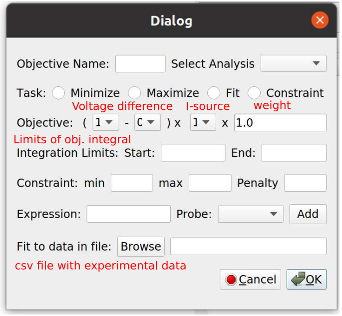
To plot all the curves stored by the current and voltage probes –

- Voltage and current probes are listed in the left and right lists.
- Click a curve in left list and click Move to right.
- Click a curve in right list and click Move to left.
- Delete : delete curves in respective lists.
- Specify min/max, labels of x-axis, left-y, or right-y axes in the text.
- `logplot` Check the boxes in ‘log’ column.
- Create Plot : To create a plot in a new window.

GUI-Side-panes - Fitting and optimization



(d) ‘Fitting/Optimization’ tab in the side-pane.



(e) ‘Fit Objective’ window to add an objective.

- Add fit parameters: Activate left Add button. Click on any component and select the fit-parameter.
- Add objective: Click on right Add button. A dialog box (see right figure in this slide) will appear. Configure optimization.
- Specify optimization algorithm and other parameters.
- Accept fitted parameters: After optimization, store optimized values as the parameters values.

To configure fit-objective –

- Select objective name, analysis, expression to be used for fitting, etc.
- In the case of fitting, provide a csv file containing experimental data corresponding to the objective.

Table of Contents

Easy-to-install licenses

For ordering a license, along with the name and the organization details, please also provide –

- For the node-locked licenses: Ethernet mac address of the client machine on which the software will run.
  OR
- For the server licenses: Ethernet mac address of the server machine at the client organization.

If you purchased one or more node-locked licenses, you will receive the following license file by secured email.

1. NodeLockedLicense_<id>_<Info>.lic, where <id> stands for license id and <Info> stands for customer identification in short.

Copy the license file to `/var/local/oesoft/licenses/` on the machine whose mac-address has been provided and change its access rights to `777`.

If you purchased one or more server licenses, you will receive the following license file by secured email.

1. ServerLicense_<id>_<Info>.lic

Copy the license file to `/usr/share/oesoft/licenses/` on the *server machine* whose mac-address has been provided.

The End

Questions? Comments?