MODELngspicer User Guide

Version 2.2

Developed by ペE

1 Introduction

MODELngspicer is a GUI application built on top of ngspice, designed to enhance workflow efficiency in circuit design optimization and device modeling. The graphical interface is developed using Python's Qt library (PySide6), allowing users to adjust model parameters via mouse wheel or numeric input and instantly visualize changes through real-time plotting.

Ngspice is an open-source SPICE-based circuit simulator that supports a wide range of analysis types, including DC, AC, transient, noise, and S-parameter analysis. It also accommodates industry-standard models such as Verilog-A and BSIM, enabling high-precision device modeling. Since it operates via command line, ngspice is well-suited for complex scripted computations and automated analysis workflows.

Ngspice official website:
https://ngspice.sourceforge.io

2 Installation

2.1 System Requirements

MODELngspicer has been tested under the following environment:

- Operating System: Windows 10 or later (64-bit)
- Python: Version 3.13 or later
- Required packages: PySide6, PyQtGraph, NumPy
- Ngspice: Version 43 or later is recommended

2.2 Installation Guide (Windows)

This section outlines the installation procedure for Windows environments. We recommend using Chocolatey, a package manager for Windows, to streamline the setup process. Alternatively, you can download and install manually from the official website.

1. Installing Chocolatey

Open PowerShell with administrator privileges and run the following command. For more details, visit Chocolatey's official installation page (https://chocolatey.org/install).

```Powershell
Set-ExecutionPolicy Bypass -Scope Process -Force;
[System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object
System.Net.WebClient).DownloadString('https://community.chocolatey.org/
install.ps1'))
```

2. Installing Python and Ngspice

To install Python and Ngspice using Chocolatey, run the following commands in an administrator PowerShell or Command Prompt. After installation, verify that **python**, **pip**, **ngspice**, and **ngspice_con** commands are available.

```Powershell
choco install python
choco install ngspice
```

3. Installing Required Libraries

Run the following command to install the required Python libraries.

```Powershell
pip install pyside6 pyqtgraph numpy
```

Once the setup is complete, you can launch MODELngspicer by executing **run.vbs** located in the project folder.

3 GUI and Basic Operations

3.1 Overview of the Interface

The following figure illustrates the GUI layout of MODELngspicer:

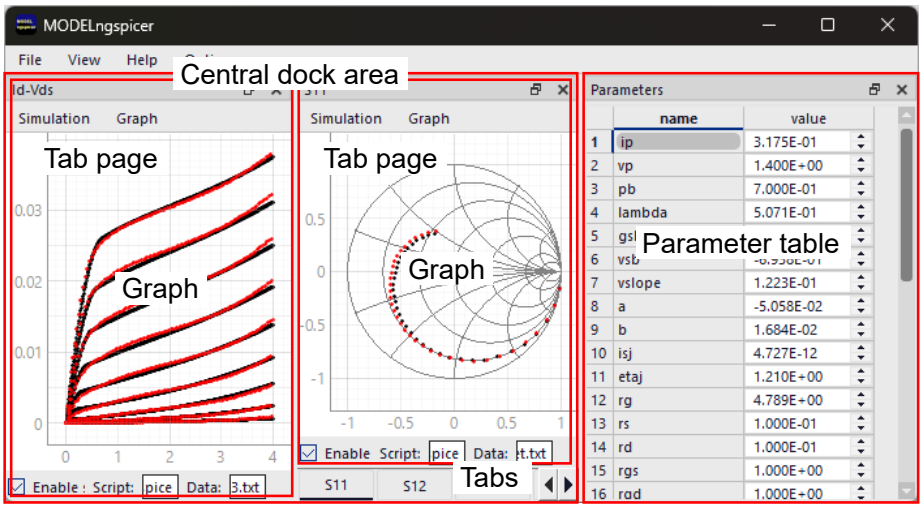

Fig. 3-1 Graphical interface

- **Central dock area**
  The area where tab pages are hosted.

- **Tab pages**
  Up to 10 tab pages may be added, each maintaining its own simulation settings. Visibility can be toggled via the **View** menu, and tab names can be edited via **Simulation** > **Rename title**.

- **Tabs**
  Tab pages are stacked within the central dock area to form tabs.

- **Graph**
  Simulation results and external data are plotted. Simulation results appear as black dots (white in dark mode), while external data is shown as red dots.

- **Parameter table**
  A list of parameters is displayed. When a parameter file is loaded via **File** > **Load params...**, the table is updated accordingly. Each parameter value is shown in a spin box and can be adjusted using the mouse wheel. Holding down the Ctrl key while scrolling enables coarse adjustments.

3.2 Basic Operation Flow

To use MODELngspicer, the user needs to prepare at least the following two files. The formats of these files are explained in the next section.

- Parameter definition file (**.txt**)
- ngspice simulation script (**.spice**, **.sp**, or **.cir**)

The basic workflow for using MODELngspicer is outlined below.

1. Load parameters
   Select a parameter file via **File** > **Load params...** . The loaded parameters will be displayed in the parameter table.
2. Select ngspice script
   Choose an ngspice script (**.spice**, **.sp**, or **.cir**) via **Simulation** > **Select ngspice script...** .
3. Select external data (optional)
   To load external data such as measurement results or target values, use **Simulation** > **Select data file...** .
4. Enable/disable simulation
   Simulation can be toggled via **Simulation** > **Enable/disable simulation** or the checkbox at the bottom left. Disabling simulation for unused tabs may reduce execution time.
5. Configure graph display
   Adjust graph settings via the **Graph** menu. Logarithmic axes, Smith charts, and polar plots are supported.
6. Adjust parameters
   Modify parameter values using the spin boxes in the parameter table:
   - Mouse wheel → fine adjustment
   - Ctrl + mouse wheel → coarse adjustment
7. Review results
   Simulation results (black dots) and external data (red dots) are displayed in the graph area.
8. Save Model Parameters
   Save the adjusted parameters via **File** > **Save params...** .

4 File Formats

4.1 Parameter File

In MODELngspicer, parameters are defined using a plain text file (**.txt**). Each line begins with a **+** symbol, as shown below:

```
Plain Text
```
```
+ is = 1e-14
+ n = 1.5
+ rs = 0.5
+ …
```

The **+** symbol is part of SPICE syntax and indicates a continuation from the previous line. By using this format, the parameter file can be seamlessly expanded into **.param** or **.model** statements within ngspice scripts.

Example 1: Expansion into a **.param** statement

```
Ngspice Script
```
```
.param
.include model.txt
```

When an **.include** statement is placed directly below a **.param** as shown in the script above, ngspice internally expands it as follows. As a result, the contents of the parameter file **model.txt** are interpreted as **.param** arguments, enabling concise integration into simulation script.

```
Ngspice Script
```
```
.param
+ is = 1e-14
+ n = 1.5
+ rs = 0.5
+ …
```

Example 2: Expansion into a **.model** statement

```
Ngspice Script
```
```
.model DIODE1 D (
.include model.txt
+ )
```

In the example above, the contents of **model.txt** are applied as model parameters for **DIODE1**. With this approach, you can define custom device models.

4.2 Simulation Result and Data Files

MODELngspicer uses a common plain text format (**.txt**) for both simulation results and external data files. The file structure is as follows:

- No header
- First column: X-axis data
- Second and subsequent columns: Y-axis data
- Delimiters: Space or tab

Example:

```
Plain Text
```

```
1e6  -3.2  -1.1
2e6  -3.5  -1.3
3e6  -3.8  -1.6
…
```

To export simulation results, use the **wrdata** command in your ngspice script. The output file must have the same name as the script, with a **.txt** extension. This naming rule enables MODELngspicer to automatically identify the result file and plot it on the graph.

Example (ngspice script):

```
Ngspice Script
```

```
set wr_singlescale
wrdata iv.txt i(v1) i(v2)
```

In this example, running **iv.spice** will produce **iv.txt**, containing the current values of **i(v1)** and **i(v2)**.

Notes:

- The result file must have the same base name as the script file; otherwise, it will not be automatically plotted.
- Data integrity is necessay. If the file contains inconsistent or malformed data, plotting may fail or produce incorrect results.

5 Use Cases

5.1 Designing an LC Bandpass Filter

This section introduces an example of designing an LC bandpass filter using MODELngspicer.

Fig. 5-1 shows the schematic diagram, with nodes labeled from **n01** to **n05**. The parameters used are **Cval1**, **Lval2**, **Cval3**, and **Lval4**. In this example, we aim to design the bandpass filter so that its center frequency is 100 MHz.

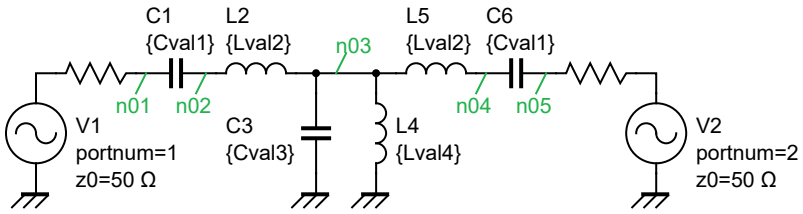

Fig. 5-1 Schematic of the LC bandpass filter

We create a parameter file (**model_initial.txt**) and an ngspice script (**bpf1.spice**), as shown below.

File: **model_initial.txt**

```
+ Cval1=10p
+ Lval2=300n
+ Cval3=253p
+ Lval4=10n
```

File: **bpf1.spice**

```
 1 bpf1.spice
 2
 3 .param
 4 .include model.txt
 5
 6 V1  n01 0       dc 0 portnum 1 z0 50
 7 V2  n05 0       dc 0 portnum 2 z0 50
 8 C1  n01 n02     {Cval1}
 9 L2  n02 n03     {Lval2}
10 C3  n03 0       {Cval3}
11 L4  n03 0       {Lval4}
12 L5  n03 n04     {Lval2}
13 C6  n04 n05     {Cval1}
14
15 .control
16 sp lin 200 50Meg 150Meg
17
18 set wr_singlescale
19 wrdata bpf1.txt db(s_2_1) db(s_1_1)
20
```

```
21 .endc
22 .end
```

Explanation of the script:
Line 1: Treated as a comment in ngspice Here, the filename is noted.
Line 3: Begins the **.param** statement.
Line 4: The **.include** directive expands parameters as arguments for **.param**.
Line 6-13: Describes the test circuit using a netlist. Voltage sources **V1** and **V2** are treated as RF ports by specifying **portnum**, and the reference impedance is set to 50 Ω using **z0**.
Line 15: **.control** starts the control section for batch processing.
Line 16: **sp lin 200 50Meg 150Meg** performs an S-parameter sweep from 50 MHz to 150 MHz with 200 linear points.
Line 18: Specifies the output format for **wrdata**: the first column is X-axis data, and subsequent columns are Y-axis data.
Line 19: Writes the S21 and S11 results to **bpf1.txt**. The filename must match the script name for automatic recognition.
Line 21: **.endc** ends the control section.
Line 22: **.end** marks the end of the script.

Launch MODELngspicer, then select **File** > **Load params...** and choose **model_initial.txt**. Next, go to **Simulation** > **Select ngspice script...** and select **bpf1.spice**.

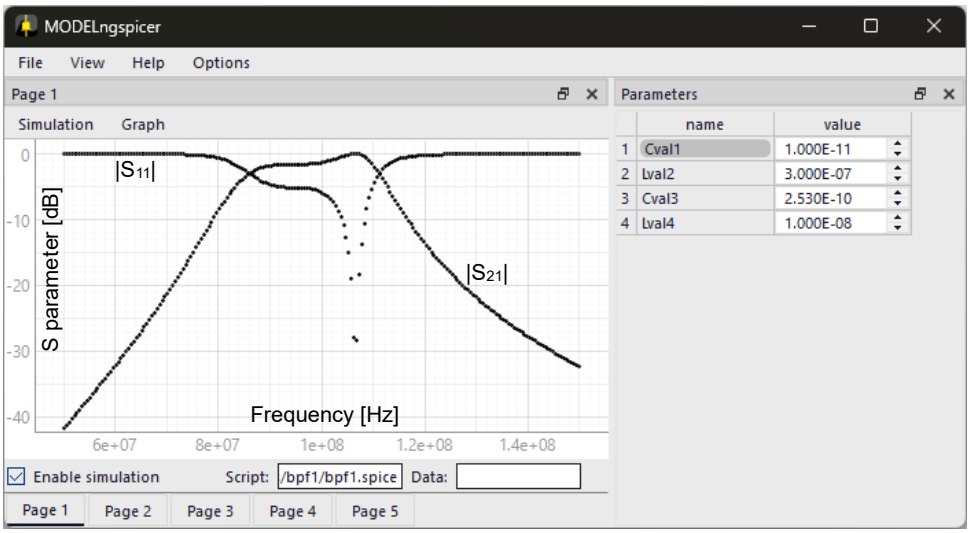

Fig. 5-2 LC bandpass filter design (Before adjustment)

After loading, the screen will appear as shown in Fig. 5-2. The graph displays the frequency response of S11 and S21. At this point, the parameters remain at their initial values, so the filter response appears slightly distorted. To improve this, try adjusting the value of **Cval1** to bring the center frequency closer to 100 MHz.

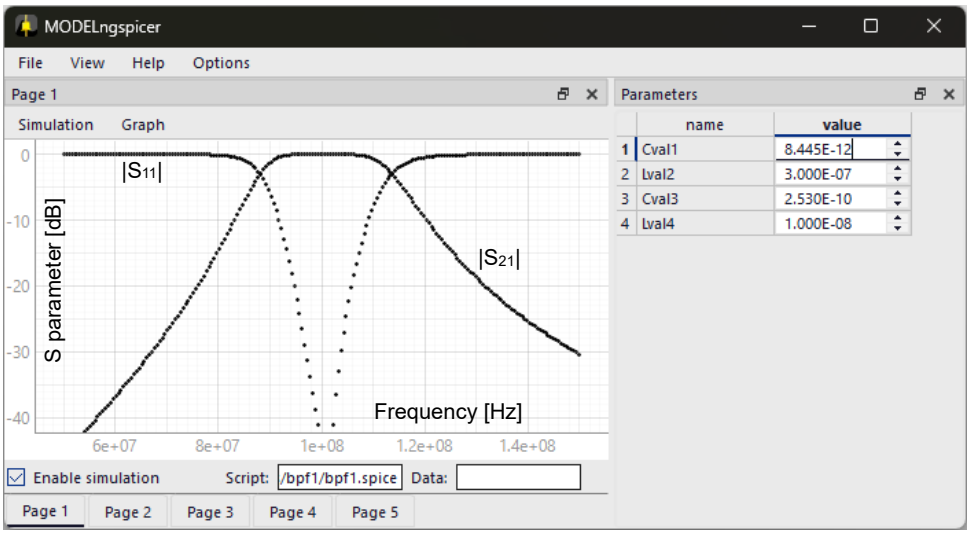

Fig. 5-3 LC bandpass filter design (After adjustment)

Fig. 5-3 shows the screen after adjustment. The response within the passband has become smoother, and the maximum gain is obtained at the center frequency. Finally, save the adjusted parameters by selecting **File** > **Save params...** from the menu bar.

5.2 Diode Modeling

This section presents an example of diode modeling using MODELngspicer. We will adjust the model parameters based on the datasheet of the Onsemi 1N4148 diode. In this example, we focus on the following characteristics for modeling.

- Forward and reverse current–voltage characteristics (I_F–V_F, I_R–V_R)
- Junction capacitance vs. reverse voltage (C_T–V_R)
- Reverse recovery time (t_{rr})

Onsemi, 1N4148 datasheet:
https://www.onsemi.com/products/discrete-power-modules/small-signal-switching-diodes/1n4148

5.2.1 Model Parameters

Below is a list of SPICE model parameters targeted for adjustment, along with the parameter definition file (**model_final.txt**).

Junction DC parameters

Name	Parameter	Units	Default	Final value
IS	Saturation current	A	1.0e-14	4.277E-09
JSW	Sidewall saturation current	A	0.0	0.000E+00
N	Emission coefficient	-	1.0	1.912E+00
RS	Ohmic resistance	Ω	0.0	7.416E-01
BV	Reverse breakdown voltage	V	infinity	1.42E+02
IBV	Current at breakdown voltage	A	1.0e-3	1.000E-03
NBV	Breakdown Emission Coefficient	-	N	4.784E+01
IKF	Forward knee current	A	0.0	0.000E+00
IKR	Reverse knee current	A	0.0	5.000E-07
JTUN	Tunneling saturation current	A	0.0	3.987E-11
JTUNSW	Tunneling sidewall saturation current	A	0.0	0.000E+00
NTUN	Tunneling emission coefficient	-	30	4.914E+02
XTITUN	Tunneling saturation current exponential	-	3	3.000E+00
KEG	EG correction factor for tunneling	-	1.0	1.000E+00
ISR	Recombination saturation current	A	1.0e-14	1.000E-14
NR	Recombination current emission coefficient	-	2.0	2.000E+00

Junction capacitance parameters

Name	Parameter	Units	Default	Final value
CJO	Zero-bias junction bottom-wall capacitance	F	0.0	8.695E-13
CJP	Zero-bias junction sidewall capacitance	F	0.0	0.000E+00

FC	Coefficient for forward-bias depletion bottom-wall capacitance formula	-	0.5	5.000E-01
FCS	Coefficient for forward-bias depletion sidewall capacitance formula	-	0.5	5.000E-01
M	Area junction grading coefficient	-	0.5	2.266E-02
MJSW	Periphery junction grading coefficient	-	0.33	3.300E-01
VJ	Junction potential	V	1.0	7.000E-01
PHP	Periphery junction potential	V	1.0	1.000E+00
TT	Transit-time	sec	0.0	4.121E-09

Temperature effects

Name	Parameter	Units	Default	Final value
EG	Activation energy	eV	1.11 (Si) 0.69 (SBD) 0.67 (Ge)	1.110E+00
XTI	Saturation current temperature exponent	-	3.0 (pn) 2.0 (SBD)	3.000E+00
TNOM	Parameter measurement temperature	°C	27	2.700E+01

Noise parameters

Name	Parameter	Units	Default	Final value
KF	Flicker noise coefficient	-	0	0.000E+00
AF	Flicker noise exponent	-	1	1.000E+00

Scaling factors

Name	Parameter	Units	Default	Final value
area	Scaling factor for area	-	1	1.000E+00
pj	Scaling factor for perimeter	-	1	1.000E+00

File: **model_final.txt**

```
+ is=4.277E-09
+ jsw=0.000E+00
+ n=1.912E+00
+ rs=7.416E-01
+ bv=1.270E+02
+ ibv=0.000E+00
+ nbv=4.784E+01
+ ikf=0.000E+00
+ ikr=5.000E-07
+ jtun=3.987E-11
+ jtunsw=0.000E+00
+ ntun=4.914E+02
+ xtitun=3.000E+00
+ keg=1.000E+00
+ isr=1.000E-14
```

```
+ nr=2.000E+00
+ cjo=8.695E-13
+ cjp=0.000E+00
+ fc=5.000E-01
+ fcs=5.000E-01
+ m=2.266E-02
+ mjsw=3.300E-01
+ vj=7.000E-01
+ php=1.000E+00
+ tt=4.121E-09
+ eg=1.110E+00
+ xti=3.000E+00
+ tnom=2.700E+01
+ kf=0.000E+00
+ af=1.000E+00
+ area=1.000E+00
+ pj=1.000E+00
```

5.2.2 Forward Current–Voltage Characteristics

We begin by adjusting the model parameters based on the forward current–voltage (I_F–V_F) characteristics. The test circuit used for this simulation is shown in Fig. 5-4, with the corresponding ngspice script (**if-vf.spice**).

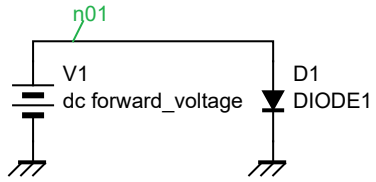

Fig. 5-4 Test circuit for forward current–voltage characteristics

File: **if-vf.spice**

```
 1 if-vf.spice
 2
 3 .model DIODE1 D (
 4 .include model.txt
 5 + )
 6
 7 V1  n01 0   dc 0
 8 D1  n01 0   DIODE1
 9
10 .control
11 option TEMP=25
12 dc V1 0.3 1.5 0.01
13
14 set wr_singlescale
15 wrdata if-vf.txt -i(V1)
16 .endc
17 .end
```

Explanation of the script:
Line 1: Treated as a comment in ngspice Here, the filename is noted.
Line 3-5: Defines the diode model **DIODE1** usgin **.model** statement. The **.include** directive expands the model parameters as arguments.
Line 7-8: Describes the test circuit using a netlist.
Line 10: **.control** starts the control section for batch processing.
Line 11: Sets the ambient temperature **TEMP** to 25°C using the **option** command.
Line 12: **dc V1 0.3 1.5 0.01** performs a DC sweep on voltage source **V1** from 0.3 V to 1.5 V in 0.01 V increments.
Line 14: Specifies the output format for **wrdata**: the first column is X-axis data, and subsequent columns are Y-axis data.
Line 15: Outputs the simulation results to **if-vf.txt**.
Line 16: **.endc** ends the control section.
Line 17: **.end** marks the end of the script.

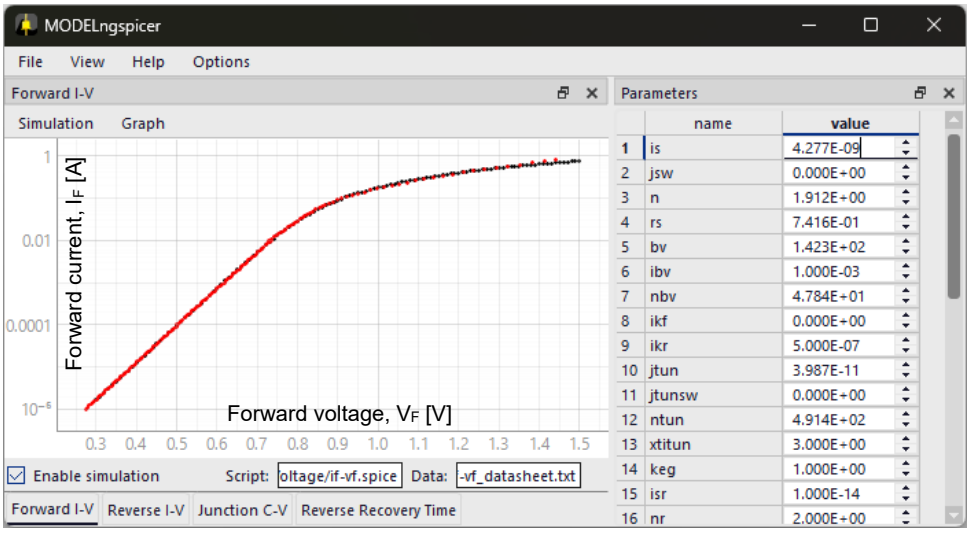

Fig. 5-5 Parameter tuning for forward I–V characteristics
(*Red dots: datasheet values, black dots: simulation results*)

Parameter tuning:
Here, we adjust the model parameters: **IS** (saturation current), **N** (emission coefficient), and **RS** (ohmic resistance). In the low-current region, **IS** and **N** affect the graph's intercept and slope, respectively. In the high-current region, **RS** becomes the dominant factor.

Fig. 5-5 shows the adjusted forward I–V characteristics. Red dots represent values extracted from the datasheet, while white dots show the simulation results. The two sets of data are in good agreement.

5.2.3 Reverse Current–Voltage Characteristics

Next, we adjust the model parameters based on the reverse current–voltage characteristics (I_R–V_R). The test circuit used for this simulation is shown in Fig. 5-6, with the corresponding ngspice script (**ir-vr.spice**).

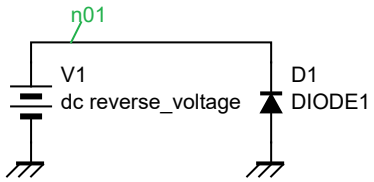

Fig. 5-6 Test circuit for reverse current–voltage characteristics

File: **ir-vr.spice**

```
1  ir-vr.spice
2
3  .model DIODE1 D (
4  .include model.txt
5  + )
6
7  V1  n01 0   dc 0
8  D1  0   n01 DIODE1
9
10 .control
11 option TEMP=25 GMIN=4e-10
12 dc V1 10 146 1
13
14 set wr_singlescale
15 wrdata ir-vr.txt -i(V1)
16 .endc
17 .end
```

Explanation of the script:
Line 1: Treated as a comment in ngspice Here, the filename is noted.
Line 3-5: Defines the diode model **DIODE1** usgin **.model** statement. The **.include** directive expands the model parameters as arguments.
Line 7-8: Describes the test circuit using a netlist.
Line 10: **.control** starts the control section for batch processing.
Line 11: Sets the ambient temperature **TEMP** to 25°C and the minimum conductance **GMIN** to 4e-10 Ω^{-1}. **GMIN** helps reproduce the diode's leakage current and is tuned to match the datasheet characteristics.
Line 12: **dc V1 10 146 1** performs a DC sweep on voltage source **V1** from 10 V to 146 V in 1 V increments.
Line 14: Specifies the output format for **wrdata**: the first column is X-axis data, and subsequent columns are Y-axis data.
Line 15: Outputs the simulation results to **ir-vr.txt**.
Line 16: **.endc** ends the control section.
Line 17: **.end** marks the end of the script.

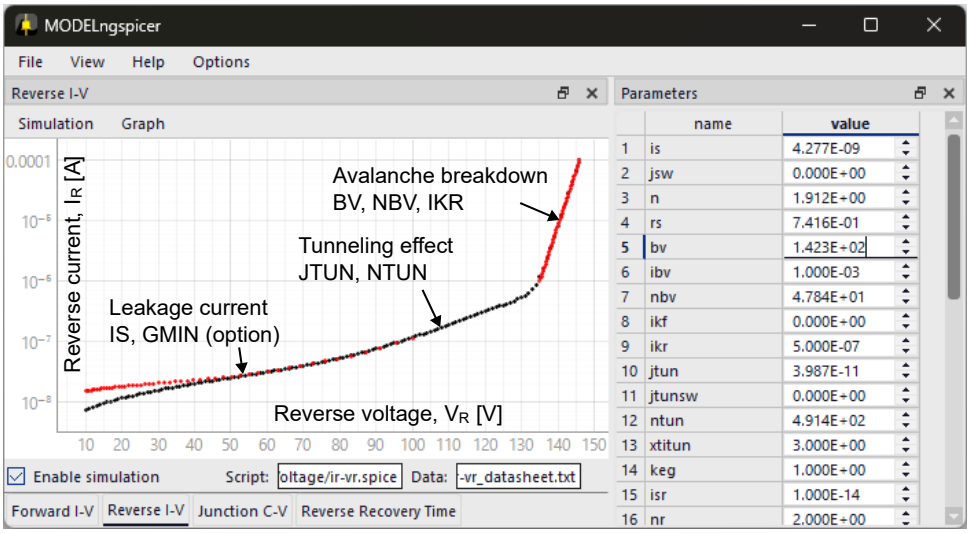

Fig. 5-7 Parameter tuning for reverse I–V characteristics
(*Red dots: datasheet values, black dots: simulation results*)

Parameter tuning:
Here, we adjust the model parameters: **JTUN** (tunneling saturation current), **NTUN** (tunneling emission coefficient), **BV** (reverse breakdown voltage), **NBV** (breakdown emission coefficient), **IKR** (reverse knee current), and the environment variable **GMIN** (minimum conductance).

- Low-voltage region (∼70 V):
  Leakage current in this region is primarily determined by **IS** and **GMIN**. Since **IS** has already been tuned based on the forward I–V characteristics, only **GMIN** is adjusted here.
- Mid-voltage region (70–130 V):
  The current slope increases in this region. To reproduce this behavior, we adjust **JTUN** and **NTUN**, which are related to tunneling current.
- High-voltage region (above 130 V):
  Avalanche breakdown dominates in this region. We adjust the breakdown voltage **BV** and the emission coefficient **NBV**. Additionally, **IKR** is set to the onset of avalanche breakdown (approximately 0.5 µA)

Fig. 5-7 shows the adjusted reverse I–V characteristics. Red dots represent values extracted from the datasheet, while white dots show the simulation results. The two sets of data are in good agreement.

5.2.4 Junction Capacitance–Reverse Voltage

Next, we adjust the model parameters based on the junction capacitance–reverse voltage characteristics (C_T–V_R). The test circuit used for this simulation is shown in Fig. 5-8, with the corresponding ngspice script (**ct-vr.spice**). A series resistor **R1** is inserted to pick up the current value during AC analysis.

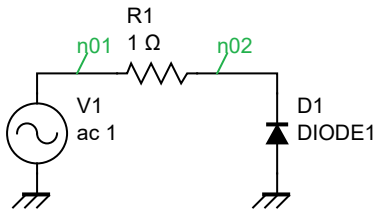

Fig. 5-8 Test circuit for C_T–V_R characteristics

File: **ct-vr.spice**

```
1  ct-vr.spice
2
3  .model DIODE1 D (
4  .include model.txt
5  + )
6
7  V1  n01 0   dc 0 ac 1
8  R1  n01 n02 1
9  D1  0   n02 DIODE1
10
11 .control
12 * Reverse voltage sweep settings
13 let st = 0
14 let sp = 15
15 let step = 0.1
16
17 let loop_index = 0
18 let loop_count = (sp-st)/step
19 let capacitance = vector(loop_count)
20 let reverse_voltage = st+step*vector(loop_count)
21
22 while loop_index lt loop_count
23     alter V1 dc reverse_voltage[loop_index]
24     ac lin 1 1Meg 1Meg
25
26     $ Calculate capacitance from admittance
27     let Y11 = v(n01,n02)/v(n02)
28     let capacitance[loop_index] = abs(imag(Y11)/(2*pi*frequency))/1e-12
29     let loop_index = loop_index + 1
30 end
31
32 setscale capacitance reverse_voltage
33 wrdata ct-vr.txt capacitance
34 .endc
35 .end
```

Explanation of the script:
Line 1: Treated as a comment in ngspice Here, the filename is noted.
Line 3-5: Defines the diode model **DIODE1** usgin **.model** statement. The **.include** directive expands the model parameters as arguments.
Line 7-9: Describes the test circuit using a netlist.
Line 11: **.control** starts the control section for batch processing.

Line 13-15: Voltage sweep settings are defined using **st** (start), **sp** (stop), and **step**.
Line 17-18: Variables **loop_index** and **loop_count** are defined for loop control.
Line 19-20: Vector variables **capacitance** and **reverse_voltage** are initialized to store output results.
Line 22: **while** loop is used to perform repeated processing.
Line 23: **alter** overwrites the value of the voltage source **V1**.
Line 24: **ac lin 1 1Meg 1Meg** performs AC analysis at a single frequency point of 1 MHz.
Line 27-28: Capacitance is calculated from admittance and stored in the **capacitance** vector.
Line 29: **loop_index** is incremented.
Line 30: marks the end of the **while** loop.
Line 32: Associates the **capacitance** and **reverse_voltage** vectors.
Line 33: Outputs the results to **ct-vr.txt**.
Line 34: **.endc** ends the control section.
Line 35: **.end** marks the end of the script.

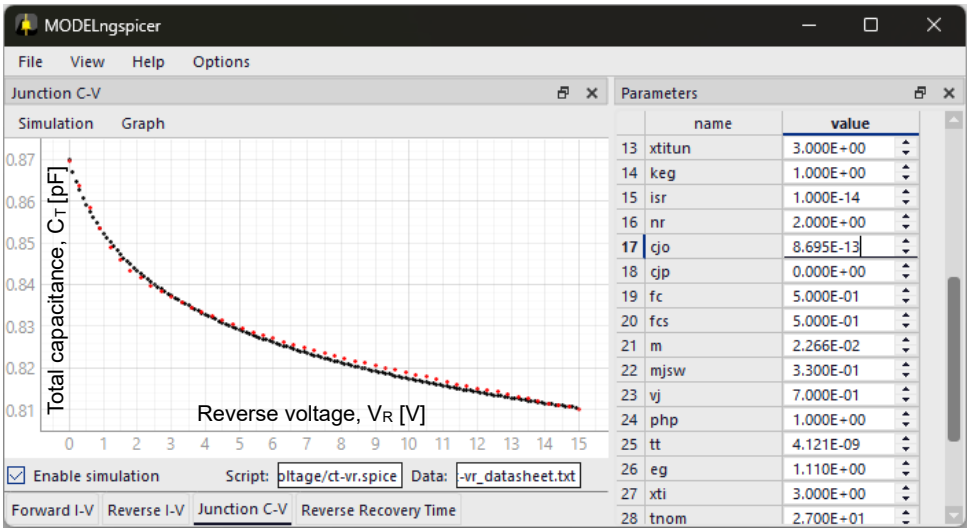

Fig. 5-9 Parameter tuning for C_T–V_R characteristics
(*Red dots: datasheet values, white dots: simulation results*)

Parameter tuning:
Here, we adjust the model parameters: **CJO** (zero-bias junction bottom-wall capacitance), **M** (area junction grading coefficient), and **VJ** (junction potential).

Fig. 5-9 shows the adjusted C_T–V_R characteristics. Red dots represent values extracted from the datasheet, while white dots show the simulation results. The two sets of data are in good agreement.

5.2.5 Reverse Recovery Time

Finally, we adjust the model parameters based on the reverse recovery time (t_{rr}). Reverse recovery time refers to the delay between switching from forward bias to reverse bias, during which excess carriers (electrons and holes) stored in the junction are swept out before the diode fully blocks current.

Typically, a measurement circuit like Fig. 5-10 is used. In this setup, a forward current I_F (~10 mA) is applied via voltage source **V2**. Then, a negative pulse is applied from the pulse generator **V1**, and the time response of the reverse current is observed until it settles.

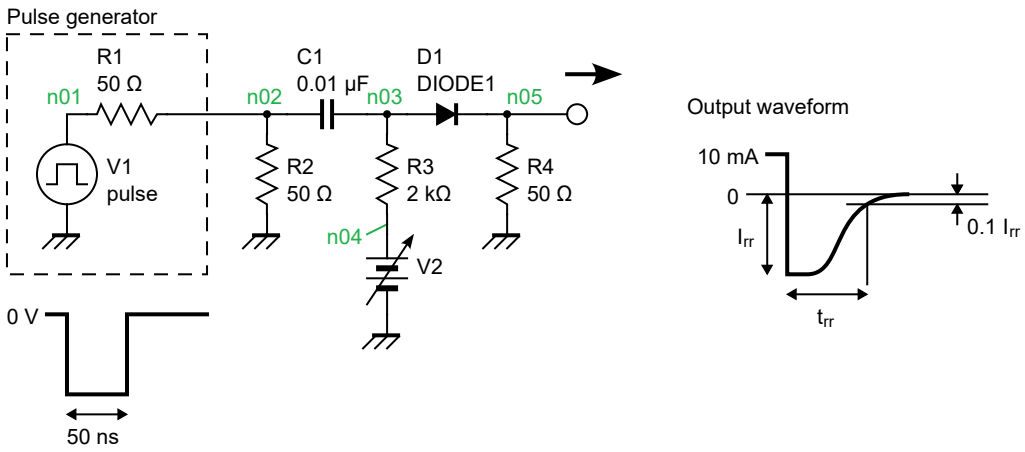

Fig. 5-10 Test circuit for reverse recovery time (t_{rr})

File: **trr.spice**

```
1  trr.spice
2
3  .model DIODE1 D (
4  .include model.txt
5  + )
6
7  V1  n01 0   dc 0 pulse(0 -3 10n 0.1n 0.1n 50n 100n 1)
8  V2  n04 0   dc 21.2
9  R1  n01 n02 50
10 R2  n02 0   50
11 R3  n03 n04 2k
12 R4  n05 0   50
13 C1  n02 n03 0.01u
14 D1  n03 n05 DIODE1
15
16 .control
17 tran 10p 20n
18 wrdata trr.txt v(n05)
19 .endc
20 .end
```

Explanation of the script:
Line 1: Treated as a comment in ngspice Here, the filename is noted.
Line 3-5: Defines the diode model **DIODE1** usgin **.model** statement. The **.include** directive expands the model parameters as arguments.
Line 7-14: Describes the test circuit using a netlist.
Line 16: **.control** starts the control section for batch processing.
Line 17: **tran 10p 20n** performs transient analysis, simulating the time response up to 20 ns with a time step of 10 ps.
Line 18: Outputs the results to **trr.txt**.
Line 19: **.endc** ends the control section.
Line 20: **.end** marks the end of the script.

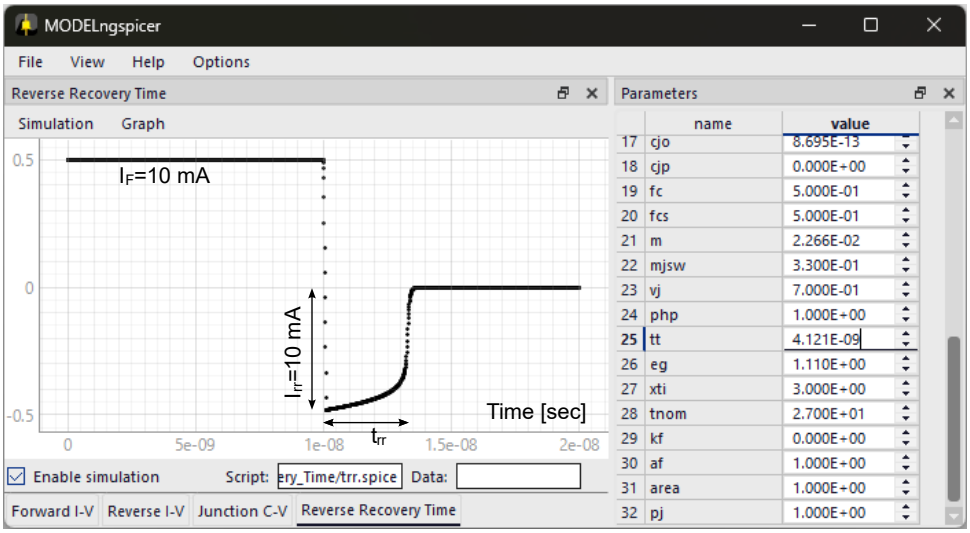

Fig. 5-11 Parameter tuning for reverse recovery time

Parameter tuning:
According to the datasheet for the 1N4148 diode, the reverse recovery time is specified as t_{rr}=3.2 ns under the condition of I_F=10 mA and I_{rr}=10 mA. To match this characteristics, the model parameter **TT** (transit time) is adjusted accordingly.

Fig. 5-11 shows the time-domain waveform of the reverse recovery response. It confirms that $t_{rr} \simeq$3.2 ns has been achieved.