ModelNgspicer User Guide

Version 1.0.0

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
https://ngspice.sourceforge.io

2 Installation

2.1 System Requirements

ModelNgspicer has been tested and confirmed to work 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. If you choose that route, make sure the executable path is properly added to your system's environment variables.

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
To install Python using Chocolatey, run the following command in an administrator PowerShell or Command Prompt:

```Powershell
choco install python
```

After installation, verify that both `python` and `pip` commands are available.

3. Installing Required Libraries
Use PowerShell or Command Prompt to install the necessary Python packages:

```Powershell
pip install PySide6 pyqtgraph numpy
```

4. Installing ngspice
Ngspice can also be installed via Chocolatey. Run the following command in an administrator PowerShell or Command Prompt:

```Powershell
choco install ngspice
```

After installation, ensure that `ngspice` and `ngspice_con` commands are available.

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:

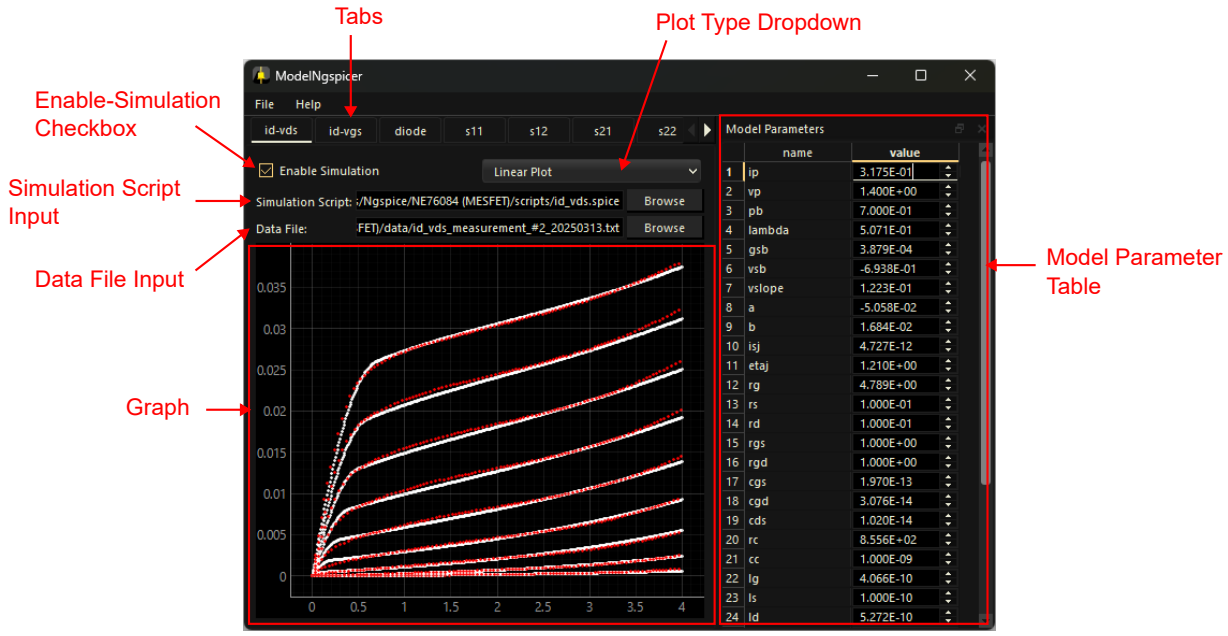

Fig. 3-1 Graphical interface

- **Tabs**
  At the top of the interface, there are ten tab panels, each allowing for different simulation configurations. You can rename a tab by double-clicking its title. Right-clicking a tab and selecting “Pop out” opens its contents in a separate window. When the pop-out window is closed, it returns to its original tab position. Multiple pop-out windows can be displayed simultaneously, enabling parallel parameter adjustments.

- **Model Parameter Table**
  This section displays a list of model parameters. You can load a parameter definition file via the menu bar: **File** > **Load Parameters...** . Each parameter value is shown in a spin box, which can be adjusted using the mouse wheel. Holding the Ctrl key while scrolling allows for coarse adjustments.

- **Enable-Simulation Checkbox**
  This checkbox toggles simulation on or off for the currently active tab. When enabled, simulations are automatically executed whenever model parameters are modified. Disabling simulations for unused tabs may help reduce execution time.

- **Simulation Script Input**
  This field allows you to select a simulation script for ngspice. Clicking the **Browse** button opens a file dialog. The selected script file (**.spice**, **.sp**, or **.cir**) is executed via the

`ngspice_con` command from Python. The results are automatically plotted on the graph below by outputting a .txt file with the same name as the script.

- **Data File Input**
  You can select a data file for comparison with simulation results. The loaded data is plotted on the same graph as the simulation output. This feature is useful for overlaying measurement results, datasheet values, or target specifications.

- **Plot Type Dropdown**
  This dropdown menu lets you choose the graph type: Linear Plot, Log-Log, Semi-Log X, Semi-Log Y, or Smith Chart.

- **Graph**
  Simulation results are plotted as white dots, while data file points appear in red.

3.2 Basic Operation Flow

The basic workflow for using ModelNgspicer is outlined below.

1. Load Model Parameters
   Click **File** > **Load Parameters...** from the menu bar and select a parameter definition file (**.txt**). The loaded parameters will apear in the **Model Parameter Table**.

2. Specify Simulation Script
   In the **Simulation Script Input** section, click the **Browse** button and select the desired ngspice script (**.spice**, **.sp**, or **.cir**).

3. Enable Simulation
   Turn on the **Enable-Simulation Checkbox** to activate simulation for the selected tab. Whenever a parameter is modified, ngspice will automatically run the simulation.

4. Adjust Parameters
   Use the spin boxes to modify model parameter values:
   - Scroll the mouse wheel for fine adjustments
   - Hold the Ctrl key while scrolling for coarse adjustments

5. Load Comparison Data (Optional)
   In the **Data File Input** section, you can specify an external data file (e.g., measurement results or target values). The data will be plotted alongside the simulation results.

6. Select Plot Type
   Use the **Plot Type Dropdown** to choose a display format: Linear, Log-Log, Semi-Log X, Semi-Log Y, or Smith Chart.

7. Review Results
   In the graph area, white dots represent simulation results, and red dots indicate external data. The graph updates in real time as parameters are adjusted.

8. Save Model Parameters
   To save the adjusted model parameters, go to **File** > **Save Parameters...** in the menu bar.

4 File Formats

4.1 Parameter Definition File

In ModelNgspicer, model parameters are defined using a plain text file (**.txt**). Each line in the file 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 the `.include` directive is placed directly under `.param`, ngspice internally expands it as follows:

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

This allows the contents of the parameter file to be treated as arguments to `.param`, enabling concise embedding within ngspice scripts.

Example 2: Expansion into a `.model` statement

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

In this case, each line in the file is applied as a parameter to the `DIODE1` model. This enables the creation of custom device models directly within ngspice.

4.2 Simulation Result and Data Files

ModelNgspicer uses a unified plain text (**.txt**) format for both simulation result files and external data files.

File Structure:

- No header row
- 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
…
```

Simulation results are generated using the `wrdata` command within ngspice scripts. The output file must have the same name as the script file, with a **.txt** extension. This naming convention allows ModelNgspicer to automatically detect and plot the results.

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:

- Filename matching is required. The result file must share the same base name as the script file; otherwise, it will not be automatically plotted.
- Data integrity matters. 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. The schematic diagram below shows the circuit, with node names labeled from `n01` to `n05`. The model parameters used are `Cval1`, `Lval2`, `Cval3`, and `Lval4`.

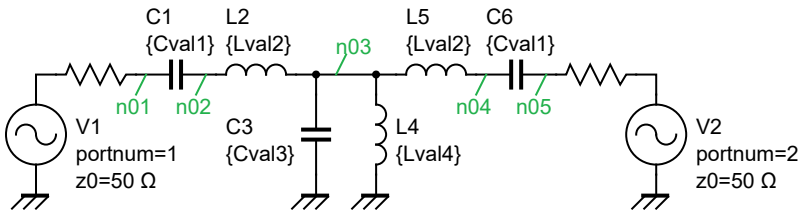

Fig. 5-1 Schematic of the LC Bandpass Filter

Let's design the filter so that its center frequency is 100 MHz. We begin by defining the initial model parameters as follows:

File: **model_initial.txt**

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

Next, we create the ngspice simulation script.

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
```

Script Breakdown:
Line 1: Treated as a comment in ngspice (used here to indicate the filename)
Line 3: Begins the `.param` block.
Line 4: Includes `model.txt`, which expands the parameter definitions.
Line 6-13: Describes the circuit using a netlist. Voltage sources `V1` and `V2` are defined as RF ports using `portnum`, with a reference impedance of 50 Ω via `z0`.
Line 15: `.control` marks the start of 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: `set wr_singlescale` ensures the output format has X-axis data in the first column and Y-axis data in subsequent columns.
Line 19: `wrdata bpf1.txt db(s_2_1) db(s_1_1)` 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.

Now that everything is prepared, let’s run the simulation using ModelNgspicer:

1. Launch ModelNgspicer.
2. From the menu bar, select **File** > **Load Parameters...** and load **model_initial.txt**.
3. Then, use the **Browse** button under **Simulation Script Input** to select **bpf1.spice**.

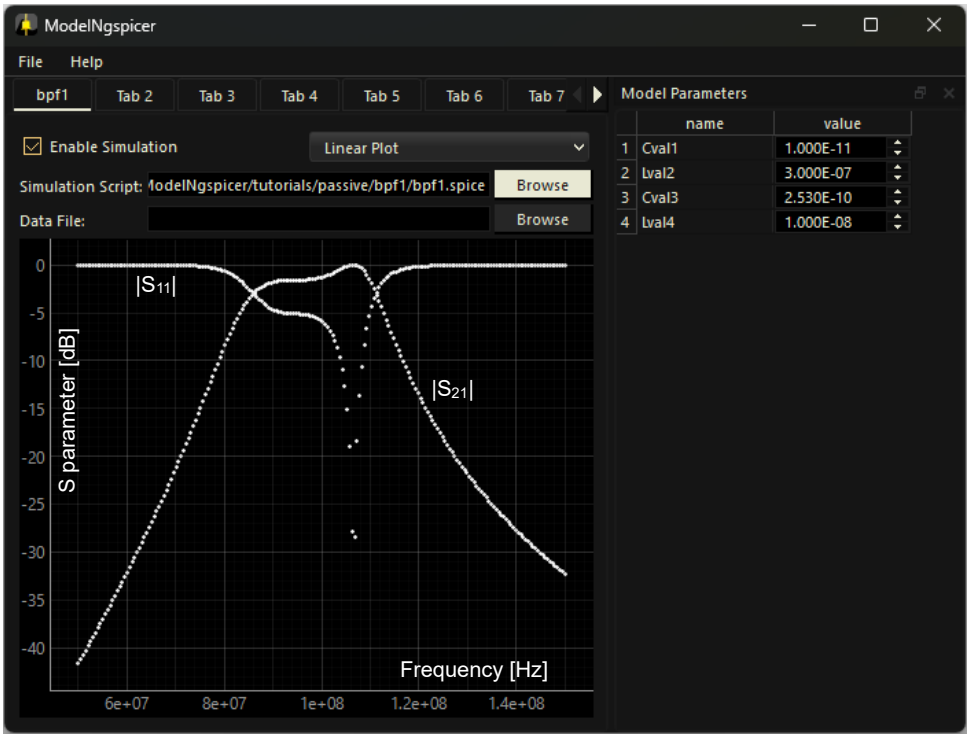

Fig. 5-2 LC Bandpass Filter, Before Tuning Parameters

The graph displays the frequency response of S21 and S11 (Fig. 5-2). At this stage, the model parameters are still at their initial values, and the filter response appears somewhat distorted.

Let’s adjust the value of `Cval1` to bring the center frequency to 100 MHz.

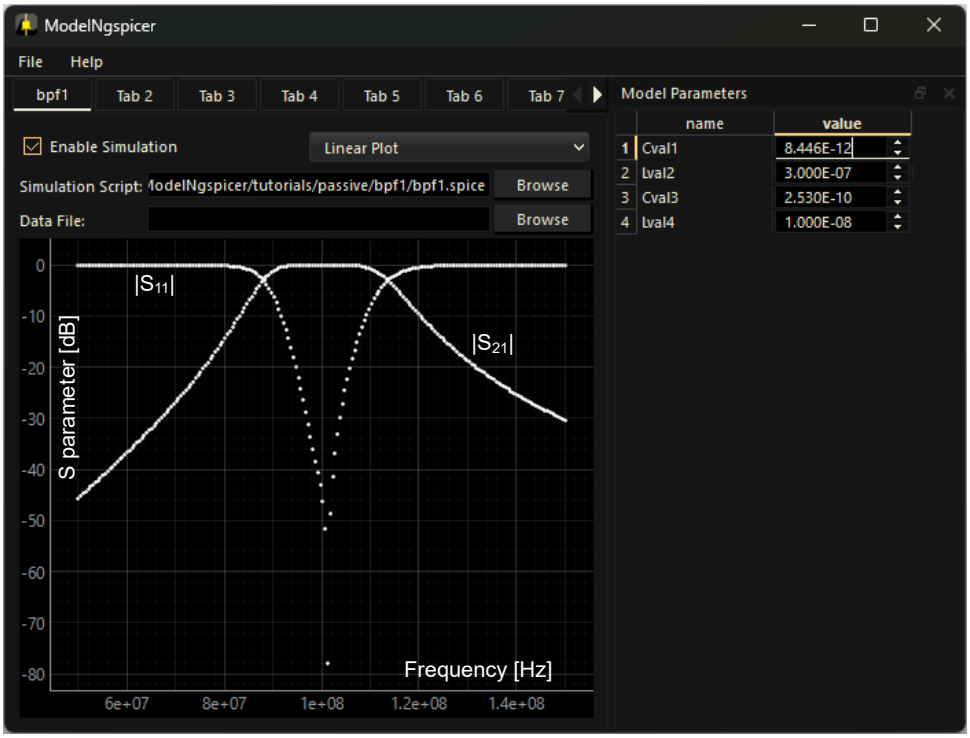

Fig. 5-3 LC Bandpass Filter, After Tuning Parameters

After tuning, the updated graph (Fig. 5-3) shows a smoother passband response, with maximum gain near the target center frequency.

Finally, save the adjusted parameters by selecting **File** > **Save Parameters...** from the menu bar. This will export the current parameter values to a text file for future use.

5.2 Diode Modeling

This section presents an example of diode modeling using ModelNgspicer. We focus on the following characteristics based on the datasheet of the 1N4148 diode from Onsemi:

- 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
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	1.432E-08
JSW	Sidewall saturation current	A	0.0	0.000E+00
N	Emission coefficient	-	1.0	2.346E+00
RS	Ohmic resistance	Ω	0.0	5.506E-01
BV	Reverse breakdown voltage	V	infinity	1.293E+02
IBV	Current at breakdown voltage	A	1.0e-3	5.000E-06
NBV	Breakdown Emission Coefficient	-	N	4.803E+01
IKF	Forward knee current	A	0.0	1.000E-02
IKR	Reverse knee current	A	0.0	1.000E-06
JTUN	Tunneling saturation current	A	0.0	6.750E-11
JTUNSW	Tunneling sidewall saturation current	A	0.0	0.000E+00
NTUN	Tunneling emission coefficient	-	30	5.238E+02
KEG	EG correction factor for tunneling	-	1.0	1.000E+00
ISR	Recombination saturation current	A	1.0e-14	1.639E-10
NR	Recombination current emission coefficient	-	2.0	1.489E+00

Junction capacitance parameters

Name	Parameter	Units	Default	Final value
CJO	Zero-bias junction bottom-wall capacitance	F	0.0	8.685E-13
FC	Coefficient for forward-bias depletion bottom-wall capacitance formula	-	0.5	5.000E-01
M	Area junction grading coefficient	-	0.5	2.289E-02
VJ	Junction potential	V	1.0	8.000E-01

TT	Transit-time	sec	0.0	4.121E-09

Temperature effects

Name	Parameter	Units	Default	Final value
EG	Activation energy	eV	1.11	1.110E+00
XTI	Saturation current temperature exponent	-	3.0	3.000E+00
TNOM	Parameter measurement temperature	°C	27	2.700E+01

Noise parameters and scaling factors

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

File: **model_final.txt**

```
+ is=1.432E-08
+ jsw=0.000E+00
+ n=2.346E+00
+ rs=5.506E-01
+ bv=1.293E+02
+ ibv=5.000E-06
+ nbv=4.803E+01
+ ikf=1.000E-02
+ ikr=1.000E-06
+ jtun=6.750E-11
+ jtunsw=0.000E+00
+ ntun=5.238E+02
+ keg=1.000E+00
+ isr=1.639E-10
+ nr=1.489E+00
+ cjo=8.685E-13
+ fc=5.000E-01
+ m=2.289E-02
+ vj=8.000E-01
+ 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
+ perimeter=1.000E+00
```

5.2.2 Reverse Current–Voltage Characteristics

We begin by tuning the model parameters to match the reverse current–voltage (I_R–V_R) characteristics. The test circuit used for this simulation is shown in Fig. 5-4, along with the corresponding ngspice script (**ir-vr.spice**).

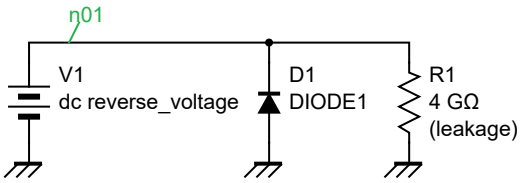

Fig. 5-4 Test Circuit for Reverse I–V Characteristics (I_R–V_R)

File: **ir-vr.spice**

```
ir-vr.spice

.model DIODE1 D (
.include model.txt
+ )

V1  n01 0   dc 0
D1  0   n01 DIODE1
R1  n01 0   4G

.control
dc V1 10 146 1

set wr_singlescale
wrdata ir-vr.txt -i(V1)
.endc
.end
```

Parameter Tuning Strategy:
To reproduce the reverse I–V characteristics, it is helpful to adjust model parameters step by step, considering the dominant physical phenomena in each voltage region:

- **Low voltage region (up to ~50 V)**:
  Tune the saturation current parameter **IS** and the high-value parallel resistor **R1** (4 GΩ) to match the diode's leakage current.
- **Mid voltage region (70-130 V)**:
  The current slope increases in this range. Adjust **JTUN** and **NTUN**, which relate to tunneling current effects.
- **High voltage region (above 130 V)**:
  Avalanche breakdown dominates. Tune the breakdown voltage **BV** and slope parameter **NBV.** Use **IKR** to match the onset of breakdown near ~1 µA.

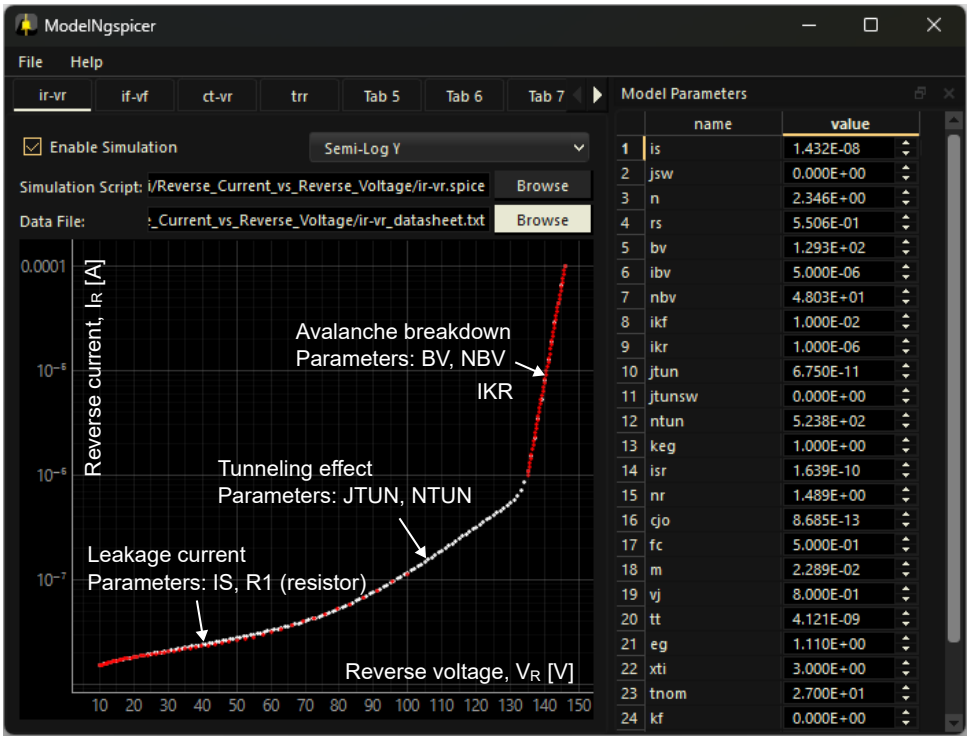

Fig. 5-5 Parameter Tuning for Reverse I–V Characteristics
(*Red dots: datasheet values, white dots: simulation results*)

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

5.2.3 Forward Current–Voltage Characteristics

Next, we tune the model parameters based on the forward current–voltage (I_F–V_F) characteristics.

The test circuit (Fig. 5-6) and simulation script (**if-vf.spice**) are shown below. To match the datasheet behavior, we adjust the following parameters: **N**, **RS**, **IKF**, **ISR**, **NR**, **XTI**.

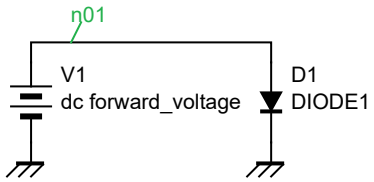

Fig. 5-6 Test Circuit for Forward I–V Characteristics (I_F–V_F)

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 foreach temp_value -40 25 65
12     option temp=$temp_value
13     dc V1 0.3 1.4 0.01
14 end
15
16 * Forward current at each temperature
17 let i1 = -dc1.i(V1)
18 let i2 = -dc2.i(V1)
19 let i3 = -dc3.i(V1)
20
21 set wr_singlescale
22 wrdata if-vf.txt i1 i2 i3
23 .endc
24 .end
```

Script Explanation:
Line 11-14: The control section uses a `foreach` loop to perform DC analysis at three temperatures: −40°C, 25°C, and 65°C. The results of each DC analysis are stored in `dc1`, `dc2`, and `dc3`, respectively.
Line 17-19: Variables `i1`, `i2`, and `i3` hold the forward current values at each temperature.

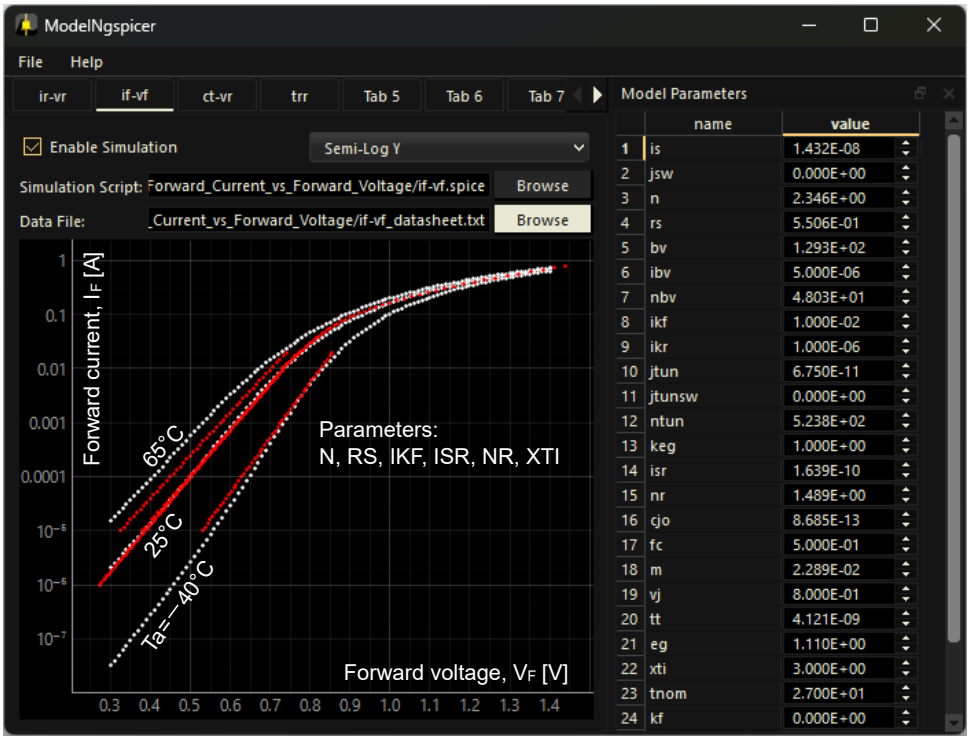

Fig. 5-7 Parameter Tuning for Forward I–V Characteristics
(*Red dots: datasheet values, white dots: simulation results*)

The adjusted forward I–V characteristics are shown in Fig. 5-7. At Ta=25°C, the simulation results closely match the datasheet. At −40°C and 65°C, slight deviations are observed, but they are considered acceptable for this model and have been left as-is.

5.2.4 Junction Capacitance vs. Reverse Voltage

Next, we tune the model parameters based on the voltage dependence of junction capacitance (C_T–V_R). The test circuit (Fig. 5-8) and simulation script (**ct-vr.spice**) are shown below. To match the datasheet characteristics, we adjust the parameters **CJO**, **M**, and **VJ**.

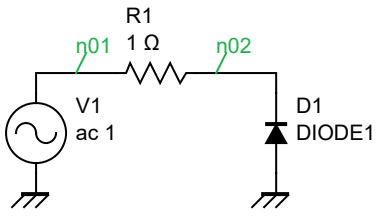

Fig. 5-8 Test Circuit for C_T–V_R Characteristics

File: **ct-vr.spice**

```
ct-vr.spice

.model DIODE1 D (
.include model.txt
+ )

```

```
 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
```

Script Explanation:
Line 22-30: A `while` loop sweeps the DC voltage of `V1` from 0 V to 15 V in 0.1 V steps. Within each iteration, an AC analysis is performed at 1 MHz, and the capacitance is calculated.
Line 32: The `setscale` command links the `capacitance` vector to `reverse_voltage`, treating the latter as X-axis data and the former as Y-axis data.

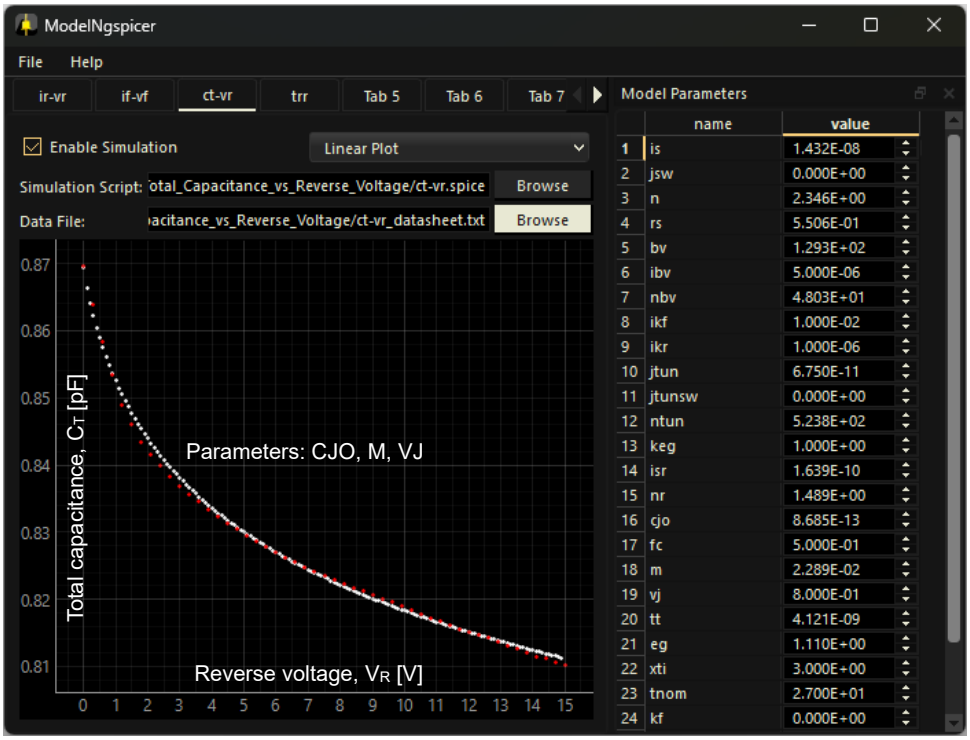

Fig. 5-9 Parameter Tuning for C_T–V_R Characteristics
(*Red dots: datasheet values, white dots: simulation results*)

The adjusted C_T–V_R characteristics are shown in Fig. 5-9. Red dots represent datasheet values, while white dots show simulation results. The two sets of data match well.

5.2.5 Reverse Recovery Time

Finally, we focus on the diode's reverse recovery time (t_{rr}) to fine-tune the model parameters. 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.

A typical test circuit is shown in Fig. 5-10. In this setup, a forward current (I_F) is established using voltage source `V2`. Then, a negative pulse is applied via `V1`, and the transient response is observed until the reverse current settles.

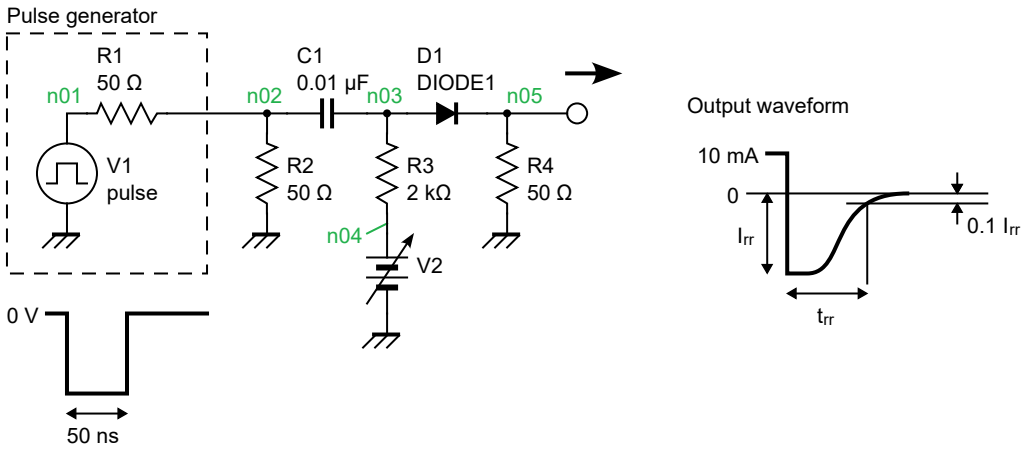

Fig. 5-10 Test Circuit for Reverse Recovery Time (t_{rr})

File: **trr.spice**

```
trr.spice

.model DIODE1 D (
.include model.txt
+ )

V1  n01 0   dc 0 pulse(0 -3 10n 0.1n 0.1n 50n 100n 1)
V2  n04 0   dc 21.2
R1  n01 n02 50
R2  n02 0   50
R3  n03 n04 2k
R4  n05 0   50
C1  n02 n03 0.01u
D1  n03 n05 DIODE1

.control
tran 10p 20n
wrdata trr.txt v(n05)
.endc
.end
```

According to the datasheet for the 1N4148 diode, under the conditions of I_F=10 mA and I_{rr}=10 mA, the reverse recovery time is specified as t_{rr}=3.2 ns. To match this behavior, we adjust the model parameter **TT**, which represents the carrier transit time.

The waveform in Fig. 5-11 shows the simulated reverse recovery response after tuning. The timing aligns well with the datasheet specification.

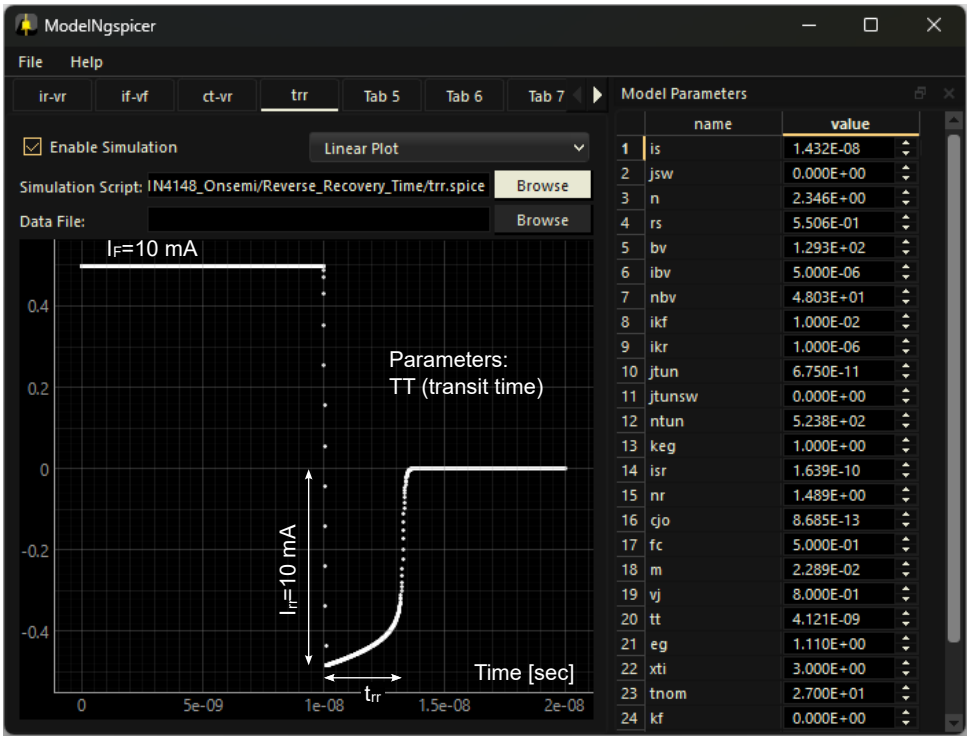

Fig. 5-11 Parameter Tuning for Reverse Recovery Time