



NL5 lite

User's Manual

Ver.3.12

VERSION

NL5 lite User's Manual version 3.12.9, 04/24/24

The latest versions of NL5 documents can be found at sidelinesoft.com/nl5.

LIMITED LIABILITY

NL5 lite, together with all accompanying materials, is provided on a “as is” basis, without warranty of any kind. The author makes no warranty, either expressed, implied, or stationary, including but not limited to any implied warranties of merchantability or fitness for any purpose. In no event will the author be liable to anyone for direct, incidental or consequential damages or losses arising from use or inability to use NL5 lite.

COPYRIGHTS

© 2024, A.Smirnov. The program and User's Manual are copyrighted. No portion of this Manual can be translated or reproduced for commercial purpose without the express written permission from the copyright holder. On publication of results obtained from use of NL5 lite citation is appreciated.



Microsoft, Windows, and Microsoft Visual C++ are registered trademarks of Microsoft Corporation.

Table of Contents

I. Introduction	4
What is NL5 lite	5
Install and run NL5 lite	5
NL5 lite file types	6
NL5 Help.....	6
NL5 preferences	7
Create and Simulate Your First Schematic.....	8
II. User Interface	11
Application.....	12
Navigation bar	12
Settings window.....	13
Context menus.....	14
Shortcuts	14
Mouse	14
Data format.....	15
Numbers	15
Names	17
Operators.....	18
Functions.....	18
Expressions	19
C language	20
III. Schematic	21
Edit schematic	22
Cursor.....	22
Wire.....	22
Ground	22
Connection	23
Component.....	23
Label	24
Zoom and scrolling	24
Selection.....	25
General commands and shortcuts	26
Components.....	27
Working with special component types and models.....	30
Subcircuit model	30
PWL model.....	33

List model (voltage and current source)	37
List model (switch and logic generator)	38
Table model	39
2D Table model.....	40
C-code component	41
DLL component	45
Roots model	49
Attachments	50
NL5 components (*.nlc)	50
Customized component.....	51
Schematic variables	52
Schematic settings	53
Schematic sheets.....	54
Check schematic.....	55
IV. Transient	56
Algorithm.....	57
Running transient	59
Transient window.....	60
Transient settings	62
Simulation	62
Add traces	65
Format traces.....	66
Screen.....	67
V. AC analysis	68
Algorithm.....	69
Running AC analysis	70
AC window	71
AC settings.....	73
Simulation	73
Add traces	75
Format traces.....	76
Screen.....	77

I. Introduction

What is NL5 lite

NL5 lite is **free** simplified version of **NL5 Circuit Simulator**. It uses the same set of components and simulation algorithm as **NL5**, and it can perform simulation with unlimited number of components. However, its user interface is simplified, and it does not support most of the features, tools, and unique functionality available in **NL5**.

NL5 lite can be used to start a new project, get initial results, and prove the concept of a new design very quickly. After that, switching to the full-function **NL5** version for detailed analysis is highly recommended.

NL5 and **NL5 lite** schematic files have the same format, so that any *.n15 schematic file can be opened and simulated by both **NL5** and **NL5 lite**. However, due to limited functionality **NL5 lite** may remove or replace schematic settings which are not supported.

For example, common changes in **NL5** schematic made by **NL5 lite** will be:

- Drawings and text in schematic/transient/AC windows removed.
- Individual symbol formatting removed.
- Formulas in component parameters replaced by values.
- Schematic Groups removed.
- Schematic variables formatting removed.
- Traces of non-supported types removed or replaced by **V** trace.
- Digital and Bus transient trace modes replaced by **Analog** mode.
- Advanced trace formatting replaced by simple formatting.
- Transient and AC data included into the file removed.
- Settings of Transient and AC Tools removed.
- And more...

Please note that all those changes will not affect simulation results.

Install and run NL5 lite

The only file required for NL5 operation is executable `n15_lite.exe`. You can place `n15_lite.exe` into any directory. There could be several copies of `n15_lite.exe` in different directories on one computer.

It is recommended to run `n15_lite.exe` with administrator privileges, to have full access to the **Registry**. To set up administrator mode in Windows 10, right-click on the NL5 icon, select **Properties**, select **Compatibility** tab, check **Run this program as an administrator**.

NL5 lite file types


The following file extensions are used by NL5 lite and NL5:

Extension	Description
n15	Schematic
n15~	Schematic backup
nlp	Preferences
n1c	NL5 component


Information about NL5 file types and icons is stored in the registry. Please run NL5 lite as administrator to be able to modify that information.

Please note that NL5 and NL5 lite schematic files are compatible and can be used by both NL5 tools. However, if you open NL5 file with NL5 lite and then save it, some NL5-specific information can be lost.

NL5 Help

NL5 lite is using the same help file as **NL5**: `n15.chm` . The help file is not required, but if used, it should be placed in the same directory as `n15_lite.exe`. The file contains only reference information, such as description of operators, functions, commands, components, and models. For detailed information refer to the NL5 lite User's Manual and NL5 User's Reference.


Please note that some information in the Help file is for NL5 only and does not apply to NL5 lite.

For context Help, press **F1**, or click **Help** button , which is available in some windows.

If you cannot see content of Help file, most likely the file is blocked. To unblock:

- Locate `n15.chm` file in the NL5 directory.
- Right-click the file, then click **Properties**.
- Select **General** tab.
- Click **Unblock**.
- If **Unblock** button is not visible, delete `n15.chm` file from the directory, copy it to NL5 directory again from `n15.zip` download package, and repeat this procedure.

NL5 preferences

Preferences  are stored in the file `n15_lite.nlp`, typically located in the same directory as `n15_lite.exe`. NL5 reads preferences from the file at start-up and saves into the file every time **Apply** or **OK** button in the **Preferences** window, and on exit.



Please note that if `n15_lite.exe` is located anywhere inside `C:\Program Files` directory, it may not be able to save preferences file `n15_lite.nlp` due to Windows writing restrictions. In this case, the preferences file will be saved in the `C:\Users\user_name\Documents\n15` directory (or similar directory, depending on specific OS).

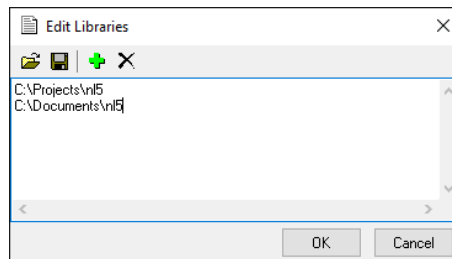
To open **Preferences** window click  in the Main menu, or in some pop-up context menus.


Some useful **Preferences** settings:

Application/Library.

Specify default library paths list for component files (**DLL**, **SubCir**, **File** model, etc.).

Click **Edit** , then enter library paths, one path per line, or click  to select from folders list.



Library path could be full path, or path relative to the directory where schematic file is located. When new schematic is created, library paths specified in Preferences will be copied into the schematic library list. To modify library paths list of a specific schematic, go to **Schematic settings** , **Library** tab.

Create and Simulate Your First Schematic

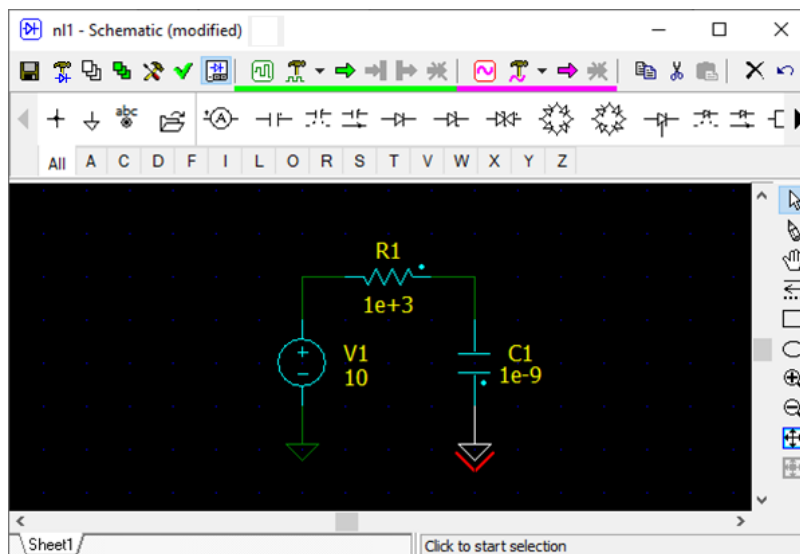
To create a new schematic, click **New** toolbar button  in the main NL5 lite window.

Enter schematic.


Entering and editing can be done using keyboard keys, mouse, or both. Here are step-by-step instructions on how to enter a simple schematic using keyboard.

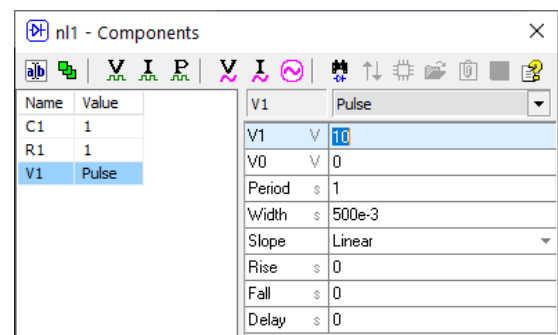
A red cursor is located in the middle of the screen and is pointing to the right.

- Press **Space** to switch to drawing mode.
- Press **Arrow Down** several times to draw short wire downward.
- Press **V** key and then press **Enter** to place a voltage source.
- Press **G** key to place a ground. Now cursor is switched back to selection mode.
- Press **Arrow Up** several times to move cursor back to the starting point.
- Press **Arrow Right** to change direction; then press **Space** to switch to drawing mode.
- Press **Arrow Right** several times to draw a short horizontal wire.
- Press **R** key and then press **Enter** to place a resistor.
- Press **Arrow Right** several times again; then press **Arrow Down** several times.
- Press **C** key and then press **Enter** to place a capacitor.
- Press **G** key to place a ground. Schematic is ready.





Edit component parameters.


- Double click on the voltage source **V1** on the schematic. A **Components** window will show up.
- Click on **V1** in the components list. Click  right to the model name (top-right), select **Pulse**.
- Click on the resistor **R1**, enter 1 in the parameter field **R**.
- Click on the capacitor **C1**, enter 1 in the parameter field **C**.

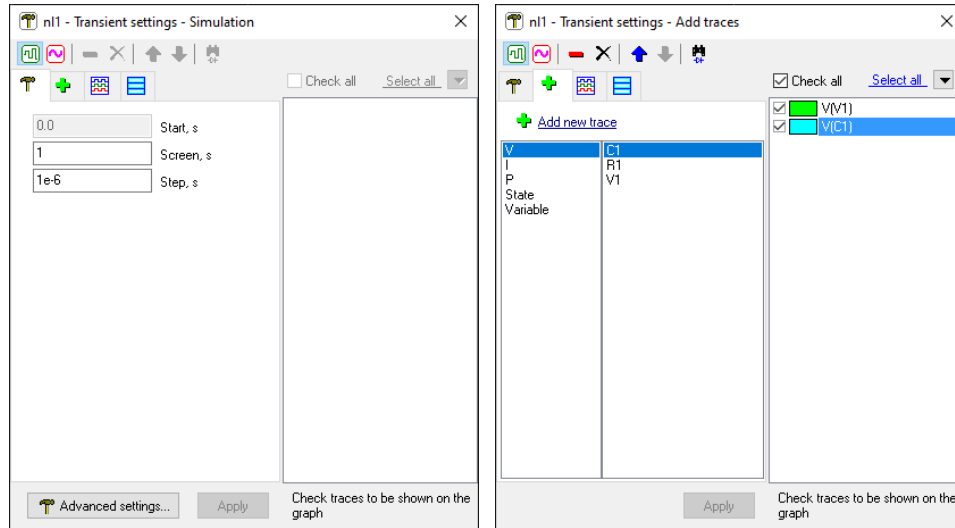


View transient settings.

Click **Transient settings**  in the schematic window toolbar, select **Simulation**  to go to **Settings** window, **Simulation** tab. You do not need to change anything here, but you can, if you wish.

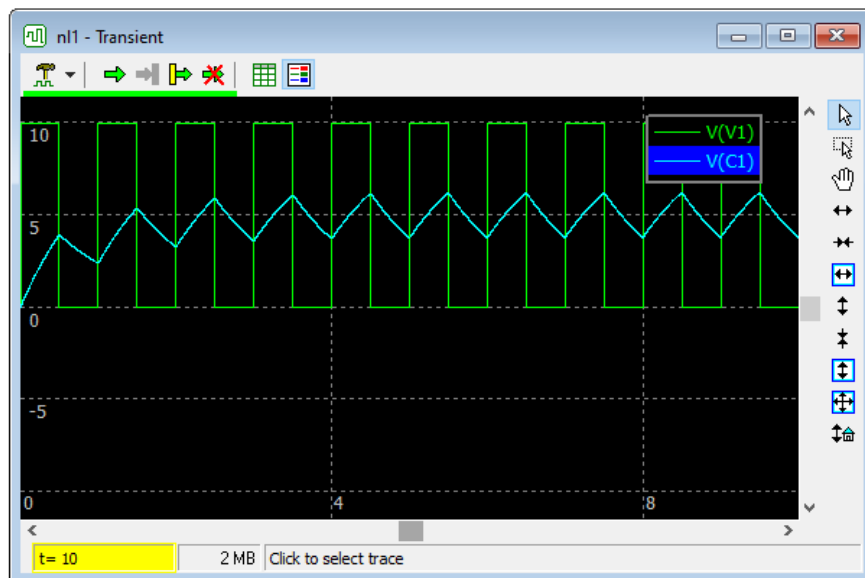
Add transient traces.

Select **Add traces** tab , select **V** in the left window, then double-click on **V1** and **C1** in the components list. Voltage traces will be added to the trace list.





Run transient.


Click **Start transient**  to run simulation:

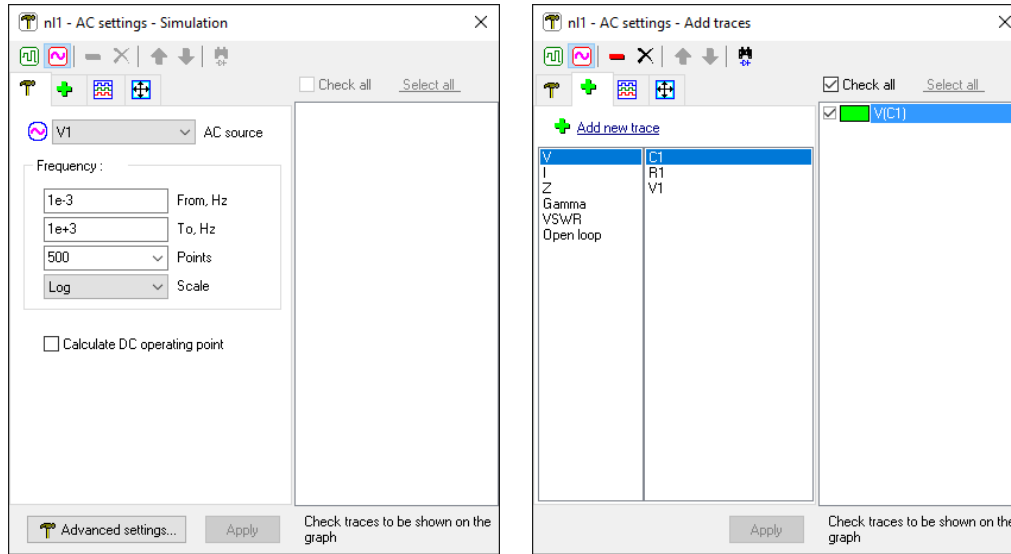


View and edit AC settings.


Click **AC settings**  in the schematic window toolbar, select **Simulation**  to go to **Settings** window, **Simulation** tab. Click on the **AC source** drop-down list, select **V1**.

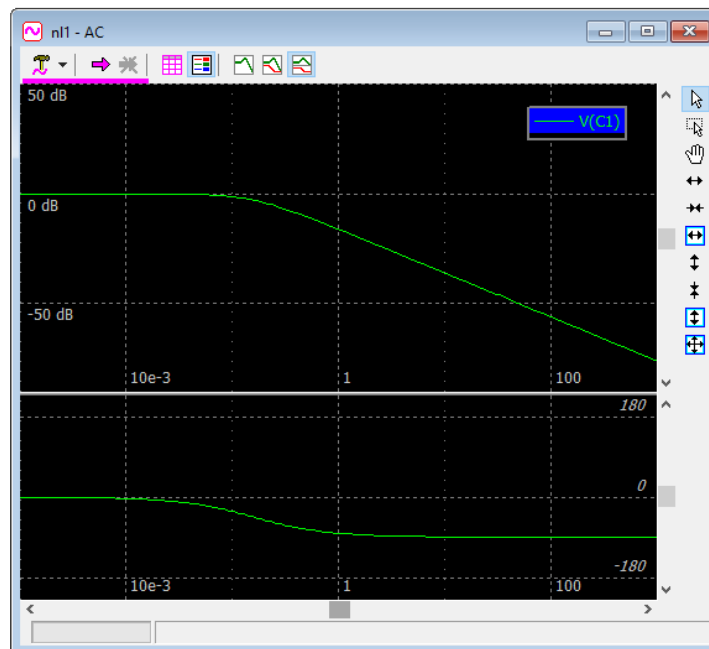
Add AC traces.

Select **Add traces** tab , select **V** in the left window, then double-click on **C1** in the components list. AC voltage trace will be added to the traces list.



Run AC.

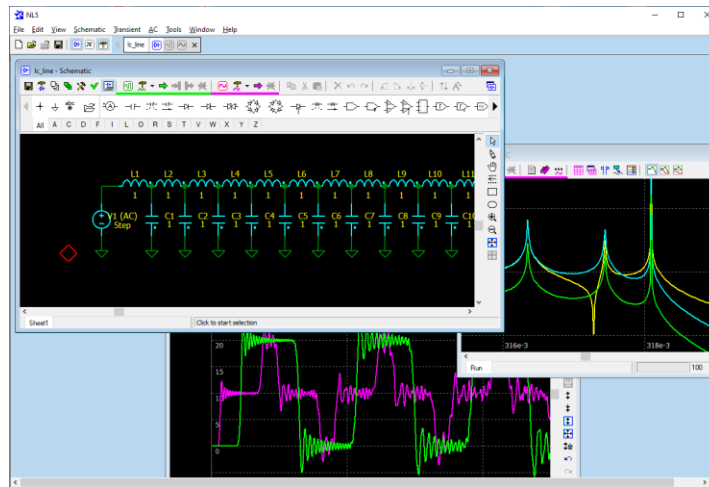
Click **Start AC**  to run AC simulation:



II. User Interface

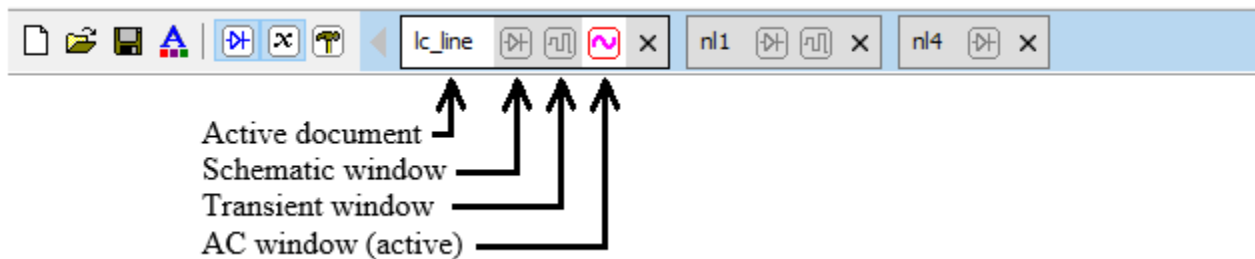
Application

NL5 lite uses standard Windows multiple-document interface (**MDI**). All schematic-related windows (Schematic, Transient, AC) are created in “MDI child” mode, and are located inside the “main” window:



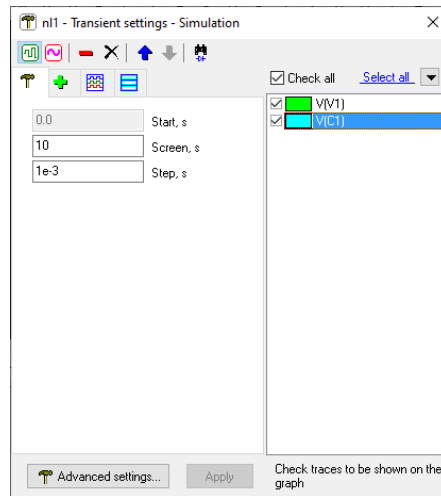
Navigation bar

Navigation bar shows currently opened documents (schematics), with highlighted active document and window:



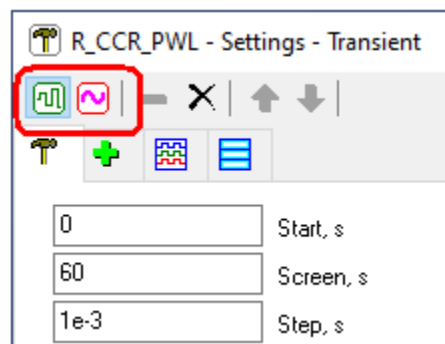
- Click on the document name to activate the document.
- Click on the window icon to activate document and window.
- Click to close schematic.

Settings window



Settings window shows settings for **Transient** and **AC** window. **Settings window** always shows information for active document (schematic). The name of active schematic is shown in the header of **Settings window**.

When **Transient** or **AC** window is activated, **Setting window** will automatically show settings for that window. You can also select desired settings clicking Transient or AC settings button (top left):



If some parameters in **Settings window** are modified, their background color will change to light-yellow:



Press **Enter** or click **Apply** to accept changes.

Settings window can be always opened.

Context menus

In some places, mouse right-click will open context pop-up menu window. The content of the context menu depends on current window, selection, position of mouse pointer, etc.

Shortcuts

- F1** – Help (context help on a specific window, selected component, C function, etc.).
- F2** – show Schematic window.
- F3** – show Components window.
- F4** – show Variables window.
- F5** – show Transient window.
- F6** – start transient.
- F7** – continue transient.
- F8** – show AC window.
- F9** – start AC analysis.
- F10** – show Settings window.

Mouse

Most of mouse and mouse wheel operations are similar in **Schematic**, **Transient**, and **AC** windows.

Some options of mouse wheel operation can be changed in **Preferences/Document/Schematic/Mouse**, and **Preferences/Document/Graphs/Mouse**.

Data format

Data format used in NL5 lite mostly complies with common engineering and scientific practice.

Numbers

Boolean number can be entered as `false` or `true` (case-insensitive):

```
bool i = true;
bool retvalue = FALSE;
```

When converted to other types, `true` is considered as 1, `false` as 0. When other types are converted to `bool`, non-zero value is considered as `true`, zero value as `false`.

Integer (`int`, `int64`) number can be entered in decimal, binary, octal, or hexadecimal formats.

Binary: use `'0b'` or `'0B'` prefix, then use digits 0 and 1:

```
0b111111111, 0B10101010, 0b10
```

Octal: start number with prefix 0 (zero), then use digits 0...7:

```
0377, 0123456
```

Hexadecimal: use `'0x'` or `'0X'` prefix. Then use 0...9, and capital or low-case A, B, C, D, E, F:

```
0xFF, 0X10aa, 0x10000
```

If a value of a number exceeds 32-bit range, it will be automatically converted to `int64` type. Use `i64` suffix to explicitly define 64-bit integer:

```
0i64, 0xfffffffffi64
```

Floating point (`float`, `double`) number can use exponential multipliers **E** or **e**, or **case-sensitive** letter multipliers:

Letter	Multiplier
T	10^{12}
G	10^9
M	10^6
k, K	10^3
m	10^{-3}
u	10^{-6}
n	10^{-9}
p	10^{-12}
f	10^{-15}

For example:

```
1.3e+3, 47E-9, 100k, 0.33u
```


Letter multiplier can be used instead of a decimal point. Zero before decimal point or letter multiplier can be omitted:

```
1k3, .47, n47
```

Infinite value is denoted by:

```
inf
```

Complex number consists of real and imaginary parts of floating point type. Imaginary part of a complex number has **lower case** letter 'j' at the end of a number. Letter 'j' cannot be used alone, only as a suffix:

```
50+45j
1+1e-3j = 1+.001j
30j
1+j : wrong! Correct format: 1+1j
```

The following **predefined constants** (case-insensitive) can be used:

```
PI = Pi = pi = 3.14159265359
RAD = Rad = rad = 180/pi = 57.2957795131
LOW = Low = low : low logical level
HIGH = High = high : high logical level
```

Constant RAD can be used to convert degrees to radians and radians to degrees:

```
degrees = radians * RAD
radians = degrees / RAD
```

where `degrees` is value in degrees, and `radians` is value in radians.

All numerical component parameters, and most of other parameters in NL5 are floating point (`double`). Those parameters can be entered in any format, however after that they are automatically converted and stored in the floating point format.

When floating point number is displayed, an engineering notation, with exponential multiplier and power of ten to be multiple of three, is used:

Entered	Displayed
1k3	1.3e+3
47e-8	470e-9
5600000	5.6e+6

Names

Component. When a new component is created, it is assigned a default name: ‘letter’ plus number:

```
R1
V2
```

To access component’s parameter in the function, use component name followed by dot ‘.’ and parameter name:

```
R1.R
V2.slope
C123.IC
```

If parameter name is not specified, a first parameter of the component will be used:

```
R1 = R1.R
C2 = C2.C
```

If current component model does not have parameters, a component model name will be used:

```
A1 = Amperemeter
```

To access a component which is part of the subcircuit, use subcircuit component name followed by dot ‘.’ and component name in the subcircuit. A nesting level is unlimited: components inside subcircuit, which in turn is part of subcircuit, can be accesses by similar notation:

```
X1.X2.V3.period
```

where x1 and x2 are subcircuits.

To access global variable of the C model of Code component x1 (in a script or command-line) use the following notation:

```
X1.variable_name
```

To access component’s model name, use component name followed by dot ‘.’ and “model”:

```
V1.model = pulse
```

If current component model does not have parameters, “.model” can be omitted:

```
S1 = Off
```

Schematic variable. Schematic variable (a variable defined in the **Variables window**) name has the same format as a component, except it does not have parameters. For example:

```
Freq
X1.var
```

Trace. The basic name of transient or AC trace consists of the letter specifying type of the trace (V, I, P), followed by component’s name in parentheses:

```
V(R1)
I(C2)
P(L3)
```

Operators

NL5 supports the following arithmetic and logical operators:

++	--	+	-	*	/	%
!	~	<<	>>	&	^	
<	<=	>	>=	==	!=	
&&		?:				
+=	-=	*=	/=	%=		
&=	^=	=	<<=	>>=		

and type-casting operators:

(bool) (int) (int64) (float) (double) (complex)

See **NL5 User's Reference** for details.

Functions

NL5 offers many standard and NL5-specific functions. The functions can be used in the C code (script, Code component) and in **Function** model of some components.

For the convenience of users, there may be several names used for the same function (for example `log10` and `lg`), so that the user can use the name he/she is more comfortable with. The following functions are available:

<code>sin</code>	<code>sqrt</code>	<code>mag, abs</code>	<code>par</code>	<code>sum</code>
<code>cos</code>	<code>sqr</code>	<code>phase</code>	<code>random, rand</code>	<code>mean</code>
<code>tan, tg</code>	<code>sq</code>	<code>re</code>	<code>gauss</code>	<code>max</code>
<code>asin</code>	<code>pow</code>	<code>im</code>	<code>limit, lim</code>	<code>min</code>
<code>acos</code>	<code>pwr</code>	<code>sign</code>	<code>islow</code>	<code>bool</code>
<code>atan</code>	<code>exp</code>	<code>round</code>	<code>ishigh</code>	<code>int</code>
<code>atan2</code>	<code>ln, log</code>	<code>floor</code>	<code>db</code>	<code>int64</code>
	<code>lg, log10</code>	<code>ceil</code>		<code>float</code>
	<code>lb, log2</code>			<code>double</code>
				<code>complex</code>

Please note that argument of functions **sin**, **cos**, **tan**, **tg** is in radians, not degrees. Similarly, functions **asin**, **acos**, **atan**, **phase** return radians, not degrees.

See **NL5 User's Reference** for details.

Expressions

Expression may consist of:

- Numbers.
- Predefined constants.
- Names of components, parameters and variables.
- Local C-code variables.
- Operators.
- Functions.
- Parentheses with unlimited nesting level.

For example:

```
2*2
2<<3
sin(2*PI*f)      // "f" is schematic variable.
max(R1,R2,R3)
1/((R1+R2)*C1)
```

Expression can be used instead of number in most entry fields in the dialog boxes, and for some component parameters. When **Enter** key is pressed, or **OK** or **Apply** button (if exists) is clicked, the expression is immediately evaluated and replaced with the numerical value.

Please note that division of two integer values will produce rounded integer result according to C-language rules **only** in expressions used in C-code component. In all other places, integer values will be considered as double values, and a result will be of double type. This is similar to how calculations are performed in calculators, Excel spreadsheets, etc. For example:

```
1/100 = 0.01   (will be 0 in C-code)
5/2 = 2.5     (will be 2 in C-code)
```

C language

Simplified C language interpreter is implemented in the NL5. It is used in the **C-code** component.

The following data types are supported:

`bool` – boolean (true/false).
`int` – 32-bit signed integer.
`int64` – 64-bit signed integer.
`float` – same as `double`.
`double` – 8-byte floating point.
`complex` – consists of `double` real and imaginary parts.

Only one-dimensional arrays are supported.

The following C keywords, statements, and operators are available:

<code>bool</code>	<code>if..else</code>	<code>continue</code>
<code>int</code>	<code>for</code>	<code>break</code>
<code>int64</code>	<code>while</code>	<code>return</code>
<code>float</code>	<code>do..while</code>	
<code>double</code>	<code>switch</code>	
<code>complex</code>	<code>case</code>	
	<code>default</code>	

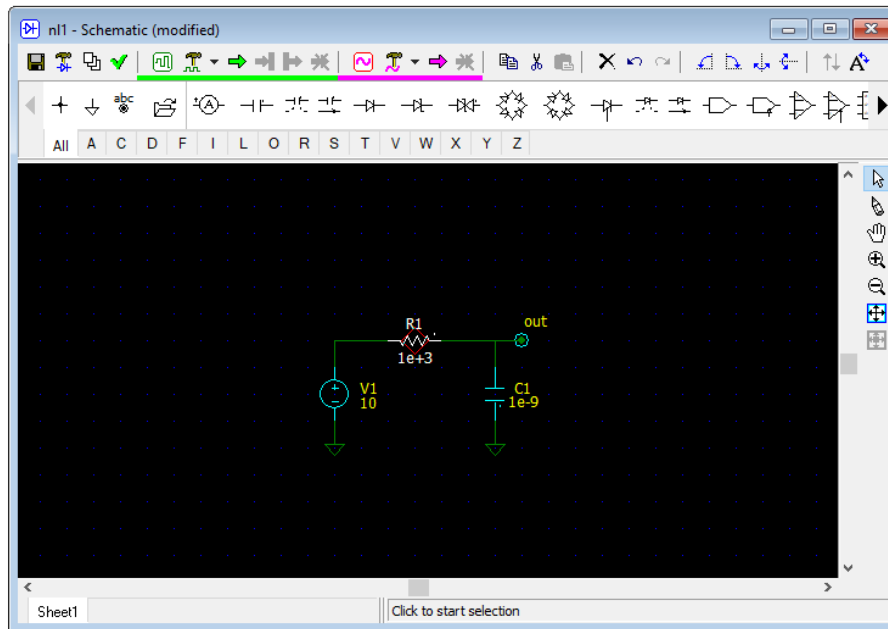
The following C language features are currently **not** implemented in NL5:

- Preprocessor (`#define`, `#include`)
- Structures and unions.
- Pointers and references.
- `goto` statement.

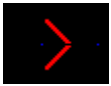
Please refer to publicly available resources for general C language syntax description and reference.

III. Schematic

Edit schematic




Cursor



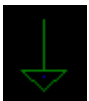
- **Double-click** - set cursor at that point and center it on the screen.
- **Left, Up, Right, Down** – change cursor direction, move cursor.
- **Click** on the cursor close to the cursor corner to change cursor direction.
- **Home** - center cursor on the screen.


Wire



- Press **Space** or click **Wire**  to switch to **Wire** mode.
- **Click** and drag to draw wire.
- **Left, Up, Right, Down** – change cursor direction, draw wire.
- Press and hold **Ctrl** to draw diagonal wire.
- Press **Space** or **ESC** to switch back to **Selection** mode.


Ground



- Press **G** or click  on the **Components bar**.

Connection








- Press '.' (dot) or click  on the **Components bar**.

Three wires coming to one point are connected automatically: a connection point will be automatically added during schematic check. Two crossing wires are not connected by default and should be connected manually. All unnecessary connection points will be automatically removed during schematic check. **Warning:** diagonal wire cannot be connected to the wires it is crossing.

Component












- Select tab with component letter on the **Components bar**, click on the component.
OR
- Press component letter key (for example 'R') one or more time to select required component.
- Move component by keyboard/mouse, rotate, mirror, flip, select view if needed.
- **Right-click** on the component to see context menu, or use toolbar buttons or shortcuts to do the following:
 -  or **Ctrl-L** – rotate component left.
 -  or **Ctrl-R** – rotate component right.
 -  or **Ctrl-M** – mirror component.
 -  or **Ctrl-F** – flip component.
 -  or +/- key – show next component view (if applicable).
- Press **Enter** to place component or press **Del** or **ESC** to cancel.

When component is placed, click on the component to select it, then use the following ways to access component editing commands:

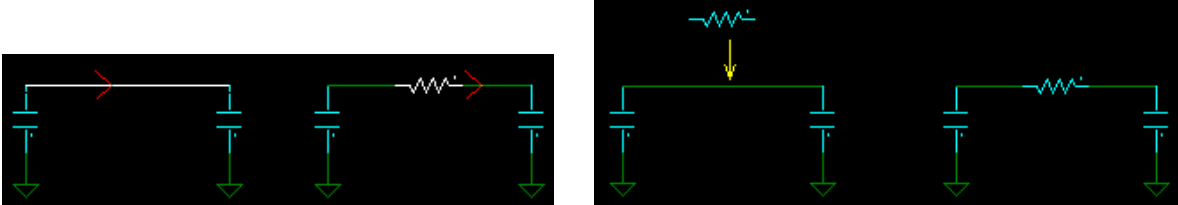
- toolbar buttons,
- keyboard shortcuts,
- **Main menu/Edit, Main menu/Edit/Selection,**
- **Right-click** on the component to see pop-up context menu.

The following component editing commands are available:

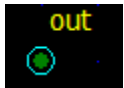
-  – set component at 45 degrees (available for some R, L, C, S, and D components).
-  or **Ctrl-T** – rotate attribute of selected component.
-  or +/- key – show next component view (if applicable).


-  or **Ctrl-L** – rotate component left.
-  or **Ctrl-R** – rotate component right.
-  or **Ctrl-M** – mirror component.
-  or **Ctrl-F** – flip component.
-  or +/- key – show next component view (if applicable).
- **Component**  : more component related commands.
- **Double-click** on the component to edit component parameters in the **Components window**. Then, if you finish editing parameters by pressing **Enter** or **Esc**, you will switch back to the schematic.

When component is placed above existing wire, a piece of the wire underneath the component is automatically removed, so that no editing of the wire is required. Similarly, when component is moved/copied above existing wire, a piece of the wire underneath the component will be automatically removed:








Label



- Press **Enter** or click  on the **Components bar**.
- For new label, enter label **Name** and **Description** (optional), click **OK**.
- For existing label, enter label **Name** or select from the list of existing labels, click **OK**.
- **Double-click** to edit label parameters in the **Components window**.

Zoom and scrolling










- Click  to switch to **Scrolling** mode, then click and drag to scroll the screen.
- OR**
- Press and hold **Shift**, then click and drag to scroll the screen.
- **Shift-Left, Shift-Up, Shift-Right, Shift-Down** - scroll the screen.
- **Ctrl-mouse wheel** - scroll horizontally.
- **Shift- mouse wheel** - scroll vertically.




-  or **PgUp** - zoom-in.
-  or **PgDo** - zoom out.
-  or **Ctrl-Home** - fit all schematic to the screen.
-  or **Shift-Home** - fit selection to the screen.
- **Mouse-wheel** – zoom in/out.

Selection



- **Click** on the schematic element to select.
- Point on empty space, **click** and drag to select a block (rectangle).
- Press and hold **Ctrl** while selecting – add to existing selection.
- **Ctrl-A** – select all schematic.
- **Right-click/Select net** - selects all wires connected to the selection either directly, or through labels (including other sheets).
- **Click** on empty space or press **Esc** to unselect (press **Esc** twice to unselect a block).
- **Left, Up, Right, Down** – move selected block.
- **Click** and drag to move element or selection.
- **Click** and drag to move attribute of selected component.
- **Shift + Click** and drag to move element or selection with **rubber bands**.
- **Ctrl + Click** and drag to copy element or selection.
- **Ctrl + Shift + Click** and drag to copy element or selection with **rubber bands**.
- **Right-click/Disable** or **Ctrl-D** to disable selection.
- **Right-click/Enable** or **Ctrl-E** to enable selection.

Disabled schematic elements are shown in “disabled” color and are not used for simulation. Disabling elements allows temporarily exclude elements from simulation without deleting.

-  or **Ctrl-C** – copy selection.
-  or **Ctrl-X** – cut selection.
-  or **Ctrl-V** – paste selection.
-  or **Del** - delete selection.
-  or **Ctrl-L** – rotate selection left.
-  or **Ctrl-R** – rotate selection right.
-  or **Ctrl-M** – mirror selection.
-  or **Ctrl-F** – flip selection.
-  - next view of selected component.

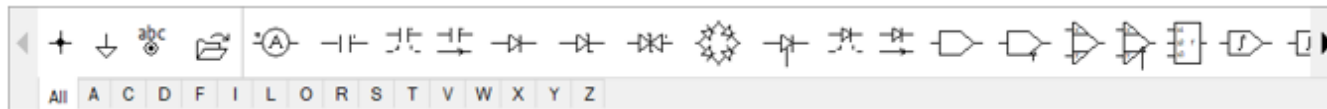
Press +/- to go through all applicable combinations of **View** , **Mirror** , and **Flip** .

General commands and shortcuts

- Press **Tab** to change attributes display for all components.
- **Right-click** to open context pop-up menu.
-  or **Ctrl-Z** – Undo.
-  or **Ctrl-Y** – Redo.

Components

All available components are shown in the **Components bar**:






Each **component** has a letter associated with it. When a new component is placed, its **name** consists of a letter and number.

Label is a special type of component: there can be many labels with the same name in the schematic. All labels with the same name are electrically connected. Labels in the subcircuit are local to the subcircuit and are not connected with labels of the same name in the main schematic or other subcircuits.

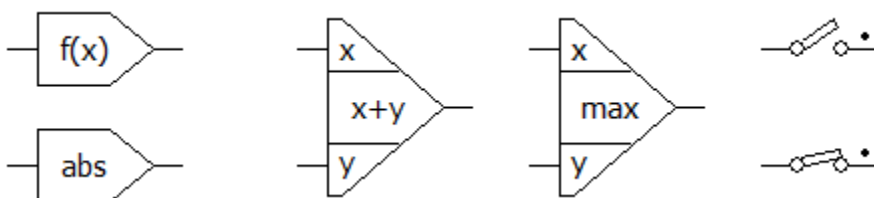
Labels can be used:

- To connect different points of the schematic without wire.
- To connect parts of the schematic located on different sheets.
- As a simulation probe (**V trace**).
- As a voltage source.

Symbol. Most of component have just one symbol associated with it, which cannot be changed. Some components may have several **views**, when the symbol is modified depending on different pin locations (for example, location of positive/negative input pins), and different functionality (for example, inversion of logical signals). For logical gates, changing **view** may also be used to change gate logical function (AND, OR, XOR).

Select component and click  to change a **view**, or press '+'/'-' keys to go through all applicable combinations of **view** , **mirror** , and **flip** .

Also, some components may have different symbols for different component **model** or component parameters. For example, symbol of the **Function** component indicates functionality of a selected model, and symbol of the **Switch** component indicates switch position in non-active state:



Model. Component **model** defines specific component functionality. For example, voltage source models include **Pulse**, **Sin**, **Step**, **File**, and more. Select component model in the drop-down list above parameters.

Parameters. Component parameters may be of different type: floating point, text, etc. Floating point parameter can be entered as an expression, for example:

```
R1.R*2.5
```

Such an expression will be immediately calculated and replaced by a resulting value.

Formula. Formula is not supported by **NL5 lite**.

Function. Some component parameters represent a **function**. Function is an expression which is recalculated at every transient or AC calculation step. Function can use the following variables:

t – current transient time, s.

f – current AC frequency, Hz.

w – angular AC frequency, $w = 2\pi f$.

s or **p** – Laplace parameter, $s = p = j*2\pi f$.

x, y – input signals for **Function** model.

V(name) – voltage on the component **name**. V trace should be available for the component.


I(name) – current on the component **name**. I trace should be available for the component.

P(name) – power on the component **name**. P trace should be available for the component.

For example:

```
sin(t*1000)*(1+cos(t*10))
(t%2>1)?1:-1
sq(V(r1))/r1
1/(1+s*R1*C1)
```

Blank parameter. Some parameters of floating point type may be **blank** (empty), which means the value is not defined. For example, blank **IC** (initial condition) parameter of capacitor means that its initial voltage at $t=0$ is not defined and will be determined during DC operating point calculation.

Logical levels and threshold for all components are defined in the **Schematic settings** , **Components** tab.

Attributes. There are 2 **attributes** of the component that can be displayed on the schematic: **Name** and **Value**.

Name is automatically assigned to the component when it is placed.

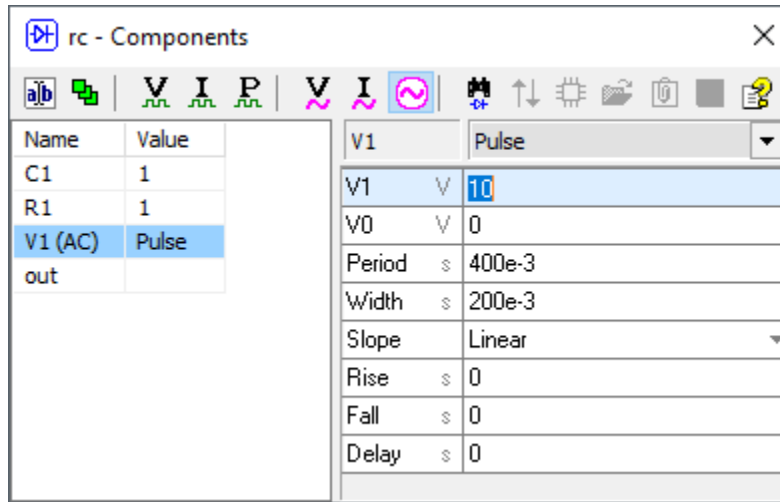
Value content depends on the component type. By default, it is the first parameter of the model, model name, or empty.

Press **Tab** to toggle attributes display mode for all components in the schematic:

- **Name** only
- **Name** and **Value**
- **Value** only
- No attributes

To move attributes, select component first, then click and drag the attribute.

Component model and parameters can be selected and edited in the **Components windows**:



Double-click on the component or press **F3** to open or switch to the **Components window**. If **double-click** on the component in the schematic window was used, edit parameter and press **Enter** to switch back to schematic window.

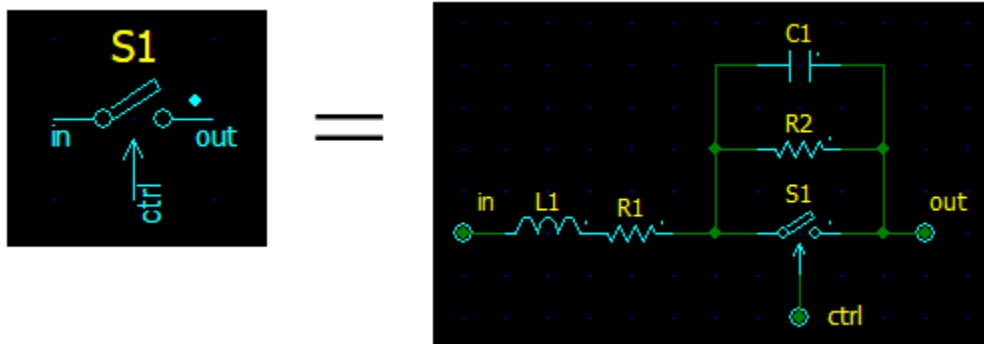
Operations available for selected component are available on the toolbar.

Working with special component types and models

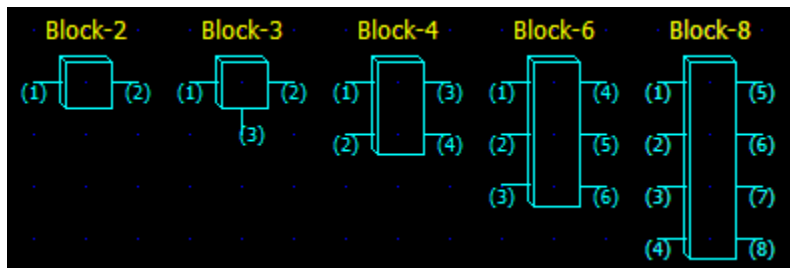
Subcircuit model


Subcircuit model (**SubCir**) allows creating simple and readable schematic by substituting some part of the schematic with one component (symbol). When simulation starts, the component with **SubCir** model is replaced by actual schematic loaded from subcircuit file.

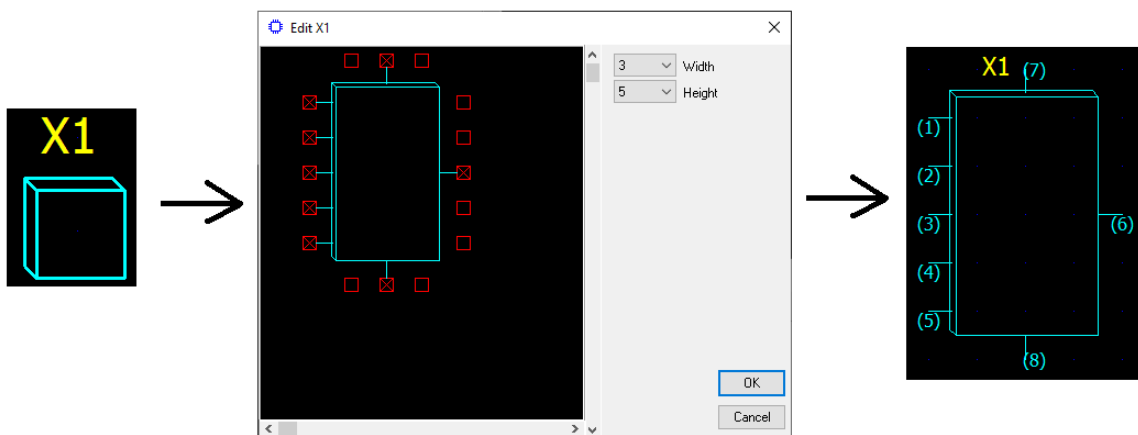
SubCir model is available almost for all components, so that some simple symbol can in fact represent quite complex circuit. For example, **switch** component with **SubCir** model may represent a complex circuit with additional resistors and parasitics:



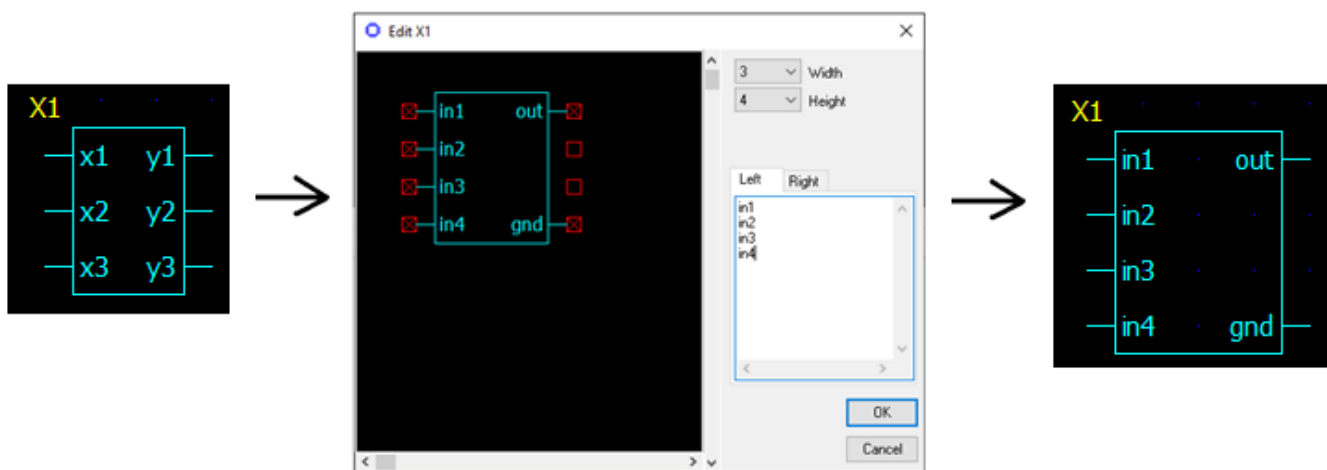
Several **X** components: **Block-2**, **Block-3**, **Block-4**, **Block-6**, and **Block-8** are offered to be used for subcircuit with fixed symbol and number of pins:



Component **X**, **Custom block** is a customized component. When placed on the schematic, it does not have any pins. Click **Edit component**  in the Components Window to open **Edit component** window:




NL5 circuit is a special component with **SubCir** model. It is a customized component as well, but in addition, names of labels connected to component pins are also defined in **Edit component** window. Them, pin names are shown on the component symbol:



SubCir model has the same parameters for all component types (except **NL5 circuit**, which does not have **Pin1...PinN** parameters):

Model	Parameter	Units	Description
SubCir	File		File name of subcircuit schematic.
	Pin1		Name of subcircuit label connected to pin 1

	PinN		Name of subcircuit label connected to pin N
	Cmd		Subcircuit start-up command string
	IC		Subcircuit Initial conditions string

File is a file name of subcircuit schematic file. Enter file name manually or click  to select the file. File extension `.nl5` can be omitted. If **File** does not have a full path, NL5 will search for the file in the following order:

- In the directory where schematic file is located.
- In the directories specified in the **Library** list (see **Schematic settings** chapter).

- In the directories specified in the **Library** list relative to the directory of schematic file.

Parameters **Pin1...PinN** define subcircuit labels connected to the pins. If label name is defined, it is displayed on the component image, otherwise a pin number in parentheses is displayed. Any of **PinN** parameters can be empty, which means that the pin is not connected to the circuit.

Cmd is a string with comma-separated expressions in “name=value” format, which is executed when subcircuit is loaded. This allows using the same subcircuit file with modified values of different components. For example:


```
R1=1k,R2=12k,C1=5n
```

where R1, R2, and C1 are subcircuit components.


IC is a string similar to **Cmd**. It defines initial conditions of subcircuit components, for example:

```
C1.IC=10,O2.IC=0
```

where C1 and O2 are subcircuit components.

IC parameter can be automatically modified by **Save initial conditions**  command: it fills in **IC** string with current IC's of all components in the subcircuit.

Subcircuit is always being loaded from the file when simulation starts, even if subcircuit is currently opened in the same NL5 instance. If subcircuit was modified, it should be saved into the file before next simulation start in order to changes take effect.

Subcircuit can be **attached**  to the component. Attached subcircuit becomes a part of the component, it will be saved into schematic file with the component, so that subcircuit file is not needed for simulation anymore. Subcircuit components may be accessed any time, even when simulation is not running.

PWL model

PWL (PieceWise Linear) model describes non-linear characteristic of the component with piecewise linear approximation.

Please note that **pwl** parameter of the **PWL** model typically specifies PWC (PieceWise Constant) function, which represents sensitivity (derivative) of PWL function. For instance, **pwl** parameter of a resistor specifies $R(V)$, which is PWC function, while resistor $I(V)$ characteristic is PWL function. The parameter is still called **pwl** (not **pwc**) for historical reasons.

The following table shows PWC (**pwl** parameter) and corresponding PWL functions for NL5 components:

Component	PWC	PWL
Resistor, diode, zener	$R(V)$	$I(V)$
Resistor	$R(I)$	$V(I)$
Capacitor	$C(V)$	$Q(V)$
Inductor	$L(I)$	$H(I)$
Voltage controlled voltage source, OpAmp	$K(V)$	$V(V)$
Current controlled voltage source	$K(I)$	$V(I)$
Voltage controlled current source	$K(V)$	$I(V)$
Current controlled current source	$K(I)$	$I(I)$

For voltage/current controlled linear components (R, C, L, and amplifier) **pwl** parameter is also PWC function, describing change of component main parameter (R, C, L, K) with control signal:

Component	PWC
Voltage controlled resistor	$R(V_{in})$
Current controlled resistor	$R(I_{in})$
Voltage controlled capacitor	$C(V_{in})$
Current controlled capacitor	$C(I_{in})$
Voltage controlled inductor	$L(V_{in})$
Current controlled inductor	$L(I_{in})$
Voltage controlled amplifier	$K(V_{in})$
Current controlled amplifier	$K(I_{in})$

PWL model of a resistor is described here as an example.

pwl parameter is a comma-separated string, describing PWC function:

$R_0, V_1, R_1, V_2, R_2, \dots, V_N, R_N$

where:

R_0 is resistance while voltage across the resistor is less than V_1 .

R_1 is resistance while voltage across the resistor is between V_1 and V_2 .

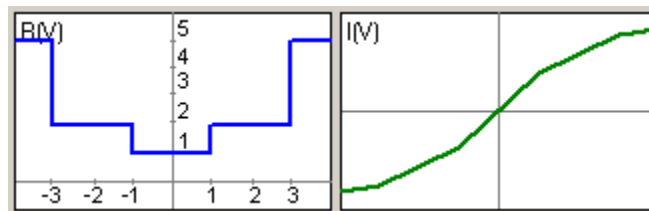
...

R_N is resistance while voltage across the resistor is greater than V_N .

Values $V_1 \dots V_N$ should be given in ascending order. Resulting PWL characteristic is calculated automatically, and always goes through the origin (0,0). Please note that only numbers can be used in **pwl** parameter: formulas are not allowed.

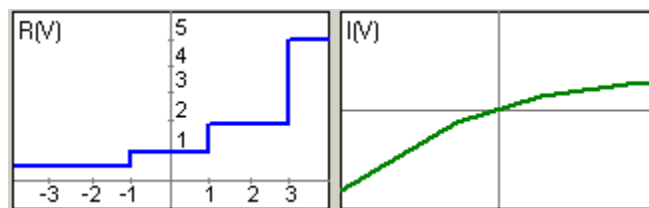
PWC function can be symmetrical or non-symmetrical. **Symmetrical** function is defined only in the interval from zero to plus infinity; the negative part of PWC function is symmetrical to positive one:

pwl = 1, 1, 2, 3, 5

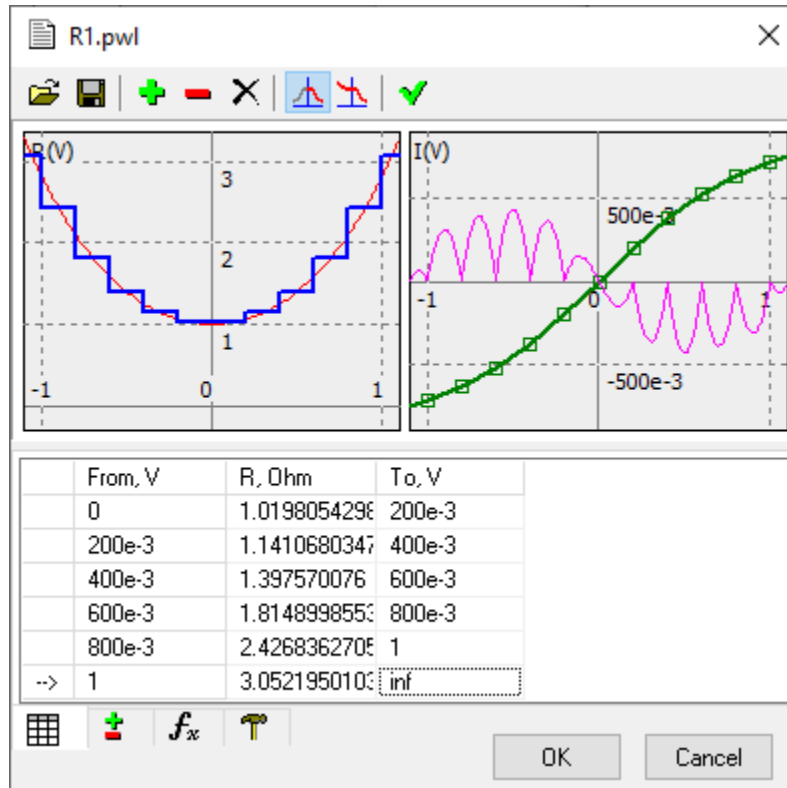


Non-symmetrical function is defined from minus to plus infinity. One of the argument points must be **zero**: it serves as an indicator of non-symmetrical characteristic:

pwl = .5, -1, 1, 0, 1, 1, 2, 3, 5




To edit **pwl** parameter, select the parameter and click  to open **PWL** window:




Right-click on the graphs area to see context menu with relevant commands.
The following tabs are available:


Table . Edit PWL data in the table.


 in the first column indicates selected row.


Select cell and edit the number: **From** value is updated automatically.


 - symmetrical PWC. The first **From** value will be **zero**.


 - non-symmetrical PWC. The first **From** value will be **-inf**.

 - split selected row.

 - remove selected row.

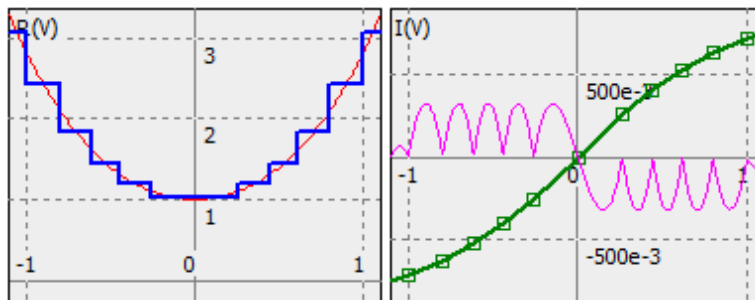
 - clear all data.

 or **Enter** - refresh graphs.

Add/remove . Add or remove points (rows) in the specified range.

Approximate f_x . Approximate arbitrary function.

- Select PWL parameter or characteristic from drop-down list. For instance, select R(V) or I(V) for resistor.
- Enter $f(x)$ as a function of parameter x .
- Click or press **Enter** to refresh graphs. Both R(V) and I(V) functions will be calculated and shown on the graph.
- To approximate function $f(x)$ using existing points (rows) in the table, fill in **To** column of the table, click **Approximate using existing points**. Select **Automatic update** to perform approximation automatically on any changes in the data table.
- To approximate function with automatic points selection to provide minimal approximation error, specify **From**, **To**, and **Interval** of approximation, click **Approximate in the range**:




List model (voltage and current source)

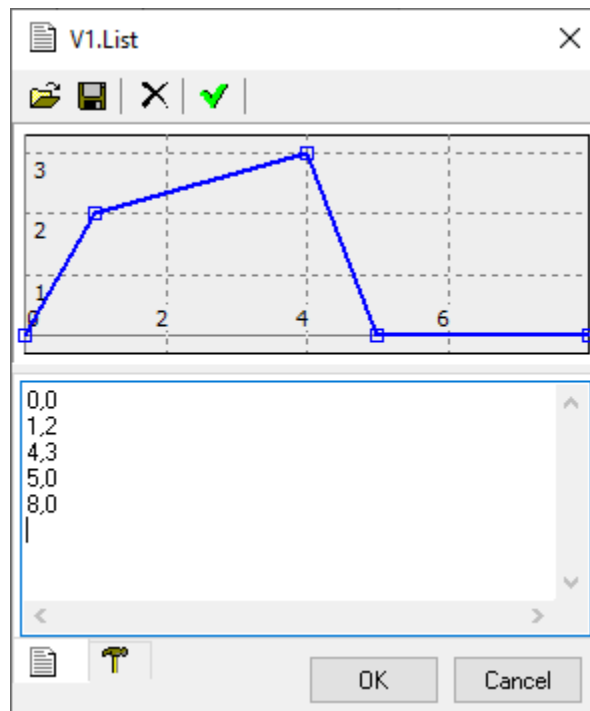
List model describes piecewise linear voltage or current source.

List parameter is a comma-separated string with time/value pairs:

$$T_1, V_1, T_2, V_2, \dots, T_N, V_N$$

Signal value between specified points is linearly interpolated. Signal value before T_1 is V_1 , signal value after T_N is V_N . Values $T_1 \dots T_N$ should be given in ascending order. Although the signal is defined on the interval $T_1 \dots T_N$, it can be repeated continuously, or delayed by setting component parameters **Delay** and **Cycle**. Please note that only numbers can be used in **List** parameter: formulas are not allowed.

Edit **List** parameter manually, or click  to open **List** window:



Enter data in the table as comma-separated time/value pairs. The data will be automatically sorted in ascending order when refreshed.

List model (switch and logic generator)

List model describes switching sequence of **Switch** component, and logical signal of **Logic Generator** component.


List parameter is a comma-separated string with time/value pairs:

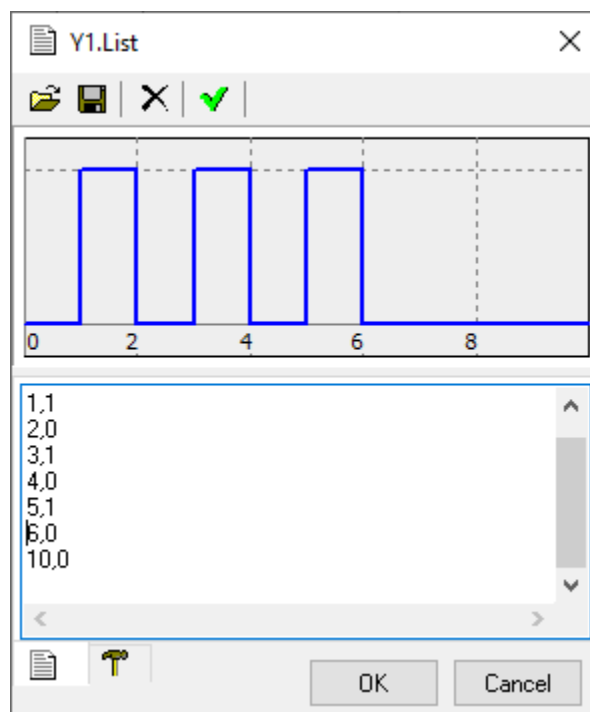
$$T_1, S_1, T_2, S_2, \dots, T_N, S_N$$

where T_i, S_i pair defines state of the signal at specified time:

- Positive value corresponds to **On** state of the switch, or **High** state of the logic generator.
- Zero or negative value corresponds to **Off** state of the switch, or **Low** state of the logic generator.

Signal state before T_1 is S_1 , signal state after T_N is S_N . Values $T_1 \dots T_N$ should be given in ascending order. Although the signal is defined on the interval $T_1 \dots T_N$, it can be repeated continuously, or delayed by setting component parameters **Delay** and **Cycle**. Please note that only numbers can be used in **List** parameter: formulas are not allowed.

Edit **List** parameter manually, or click  to open **List** window:



Enter data in the table as comma-separated time/value pairs. The data will be automatically sorted in ascending order when refreshed.


Table model

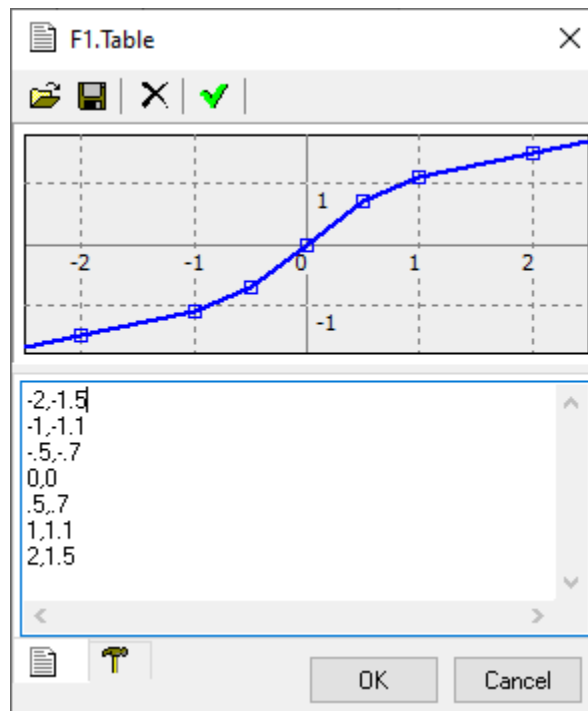
Table model describes look-up table of the **Function** component.

Table parameter is a comma-separated string with input/output pairs:

$$X_1, Y_1, X_2, Y_2, \dots, X_N, Y_N$$

where x_i, y_i pair defines input value x and output value y . Output value between specified points is linearly interpolated. Output value below x_1 is linearly extrapolated using $x_1 \dots x_2$ interval data, output value above x_N is linearly extrapolated using $x_{(N-1)} \dots x_N$ interval data. Values $x_1 \dots x_N$ should be given in ascending order. Please note that only numbers can be used in **Table** parameter: formulas are not allowed.

To edit **Table** parameter, select the parameter and click  to open **Table** window:



Enter data in the table as comma-separated input/output pairs. The data will be automatically sorted in ascending order when refreshed.

2D Table model

Table model describes 2D (two-dimensional) look-up table of the **Function-2** component.


Table parameter of the model defines output value **z** as a function of **x** and **y** inputs of the component in the following format:

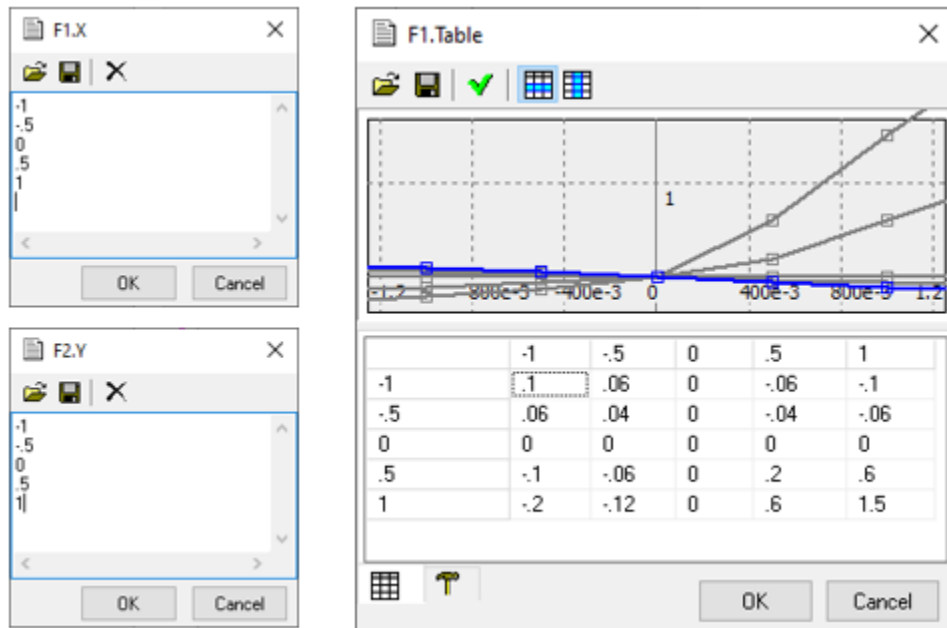
$$z_{11}, z_{12}, \dots, z_{1N}, z_{21}, z_{22}, \dots, z_{2N}, \dots, z_{M1}, z_{M2}, \dots, z_{MN}$$

where:

- z_{ij} defines output of the function for input values x_i and y_j .
- n is total number of **x** points, defined by **X** parameter of the component.
- m is total number of **y** points, defined by **Y** parameter of the component.

Output value between specified **x** and **y** points is linearly interpolated on both coordinates. Output value below x_1 is linearly extrapolated using $x_1 \dots x_2$ interval data, output value above x_N is linearly extrapolated using $x_{(N-1)} \dots x_N$ interval data. The same rule is applied to **y** coordinate. Please note that only numbers can be used in **Table**, **X**, and **Y** parameters: formulas are not allowed.


First, enter **X** and **Y** parameters, then enter **Table** values (select the parameter and click ). Columns correspond to **X** parameter, rows – to **Y** parameter:




The image shows three windows from the software interface. On the left, there are two windows for entering parameters: 'F1.X' and 'F2.Y'. Both windows have a list of input values: -1, -0.5, 0, 0.5, 1. On the right is a larger window titled 'F1.Table'. It features a graph at the top with a grid and a blue line representing a function. Below the graph is a table with the following data:

	-1	-0.5	0	0.5	1
-1	.1	.06	0	-.06	-.1
-0.5	.06	.04	0	-.04	-.06
0	0	0	0	0	0
0.5	-.1	-.06	0	.2	.6
1	-.2	-.12	0	.6	1.5

The table editor also includes a grid icon, a text tool icon, and 'OK' and 'Cancel' buttons.

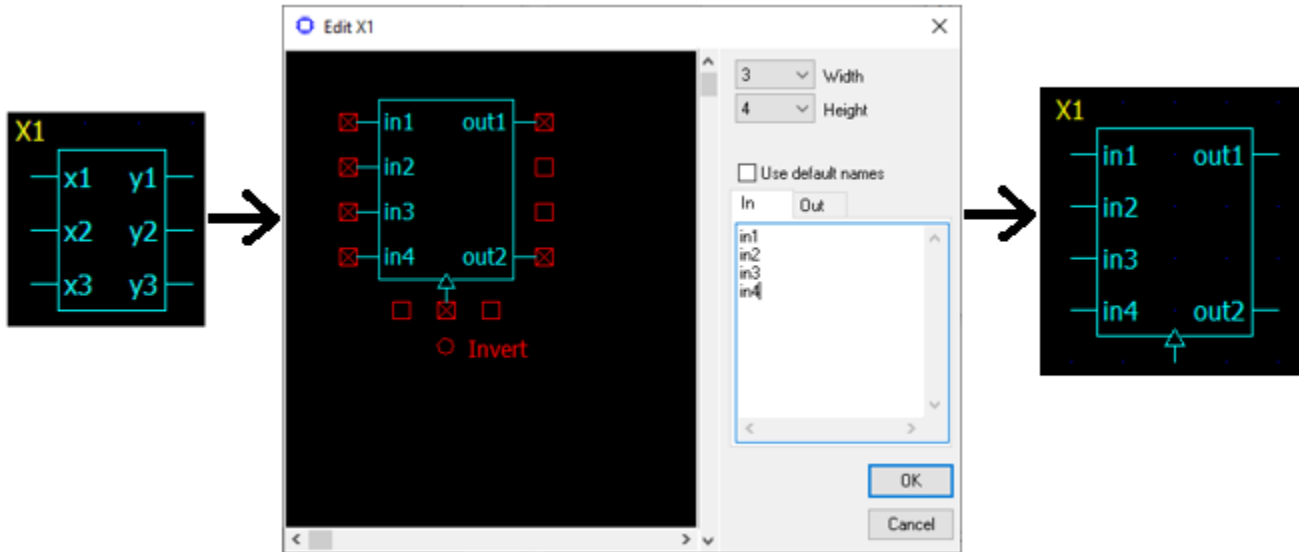
 - show $z(x)$ traces: each trace represents one row of the table. Total number of traces is m , highlighted trace corresponds to the selected cell of the table.

 - show $z(y)$ traces: each trace represents one column of the table. Total number of traces is n , highlighted trace corresponds to the selected cell of the table.

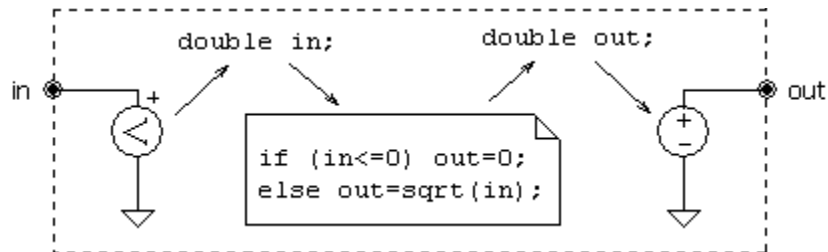
C-code component

In **C-code** component, the component function code can be written in C language. The code is interpreted by NL5 during transient simulation. Although C code interpretation is relatively slow, using **C-code** component allows fast iterations of the code and easy access to C-code variables for debugging. When the code is finalized, it can be compiled and built as DLL for faster simulation (see *Working with DLL* chapter).

C-code component is a customized component. Click **Edit component**  in the Components Window to open **Edit component** window, then specify symbol size, inputs, outputs, and clock (optional):

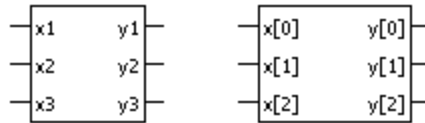


Execution. A principle of C code execution is shown on the following diagram:



Each input/output pin with a name has a variable of the same name in the C code associated with the pin. A voltage on the input pin of the component (for example, **in**) is measured by a “virtual” voltmeter, and is assigned to the variable `double in`. While C code is executed, a new value of output variable `double out` is evaluated. A voltage equal to `out` value is set to a “virtual” grounded voltage source, connected to the output pin of the same name **out**.

Inputs and outputs can also be assigned to array elements. For example, inputs `x[0]`, `x[1]`, and `x[2]` correspond to the array `double x[3]`.



Input/output variables corresponding to inputs and outputs (arrays) are **global** variables and should be declared as variables of `double` type in the beginning of the code, outside of `init` or `main` functions. (See **global variables** chapter below for details). If not explicitly declared, all required input/output variables will be automatically created during initialization phase of the analysis at $t=0$.

Code structure. A typical code structure is the following:

```
// Global variables
...

// Initialization function
init()
{
    ...
}

// Main function
main()
{
    ...
}
```

If complex initialization of global variables is not required, initialization function can be omitted. Then the code will look like this:

```
// Global variables
...

// Main function
main()
{
    ...
}
```

If global variables (other than inputs and outputs) are not used, both initialization and main functions can be omitted, and the C-code will look like that:

```
// The code
...
```

Global variables. Global variables are **public** variables of the code. They can be accessed outside the C code: from the script (for example, for data logging), from the command line, displayed as a trace on the transient graph, used in the expressions for V/I sources, etc. Also, global variables are **static**: they persist and keep their values during entire transient simulation, between calls of C-code functions.

Global variables can be used for storing code data that are calculated once during initialization, and then used in the `main` function on each simulation step. Global variables can also be used for storing values that are calculated on one simulation step, and then are used on the next simulation step.

Global variables should be declared in the beginning of the code outside of `init` and `main` functions. Variables can also be initialized during declaration. However, if some code is required for initialization, it should be done in the `init` function.

Although input/output variables are always global, they can be explicitly declared as global variables as well. If not explicitly declared, all required input/output variables will be created automatically during initialization phase of the analysis at $t=0$.

Initialization function is executed once at the beginning of simulation at $t=0$. The name of the initialization function should be `init`. Initialization code is used to initialize global variables if some complex code is required for initialization. Initialization function is optional and can be omitted. Variables declared in the `init` function are **local** and persist only during execution of that function.

Main function calculates output variables using current values of input variables during transient simulation. If **clock** pin does not exist, the function is executed on every calculation step. If **clock** pin exists, the function is executed only on rising or falling edge of logical clock signal. The name of the main function should be `main`.

Variables declared in the `main` function are **local** and persist only during execution of that function. Use global variables for storing values that are calculated on one simulation step, and then are used on the next simulation step.

Along with global and local variables, the following **read-only** variables and functions can be used in the main function:

- Component parameters (such as `R1`, `C2`, `V.period`, etc.),
- t - current time
- V(name)** - voltage on the component *name*
- I(name)** - current through the component *name*
- P(name)** – power on the component *name*
- S(name)** – state of the component *name*


where *name* is the name of any component in the schematic, and V, I, P, or S value is available for that component.

Code of `main` function can be modified during transient simulation: pause the transient, make changes, and continue transient. The changes will take effect immediately.


Initial conditions (IC parameter). Initial conditions is a string with comma-separated expressions in “name=value” format assigning initial values to output variables and global variables of the code. For example:

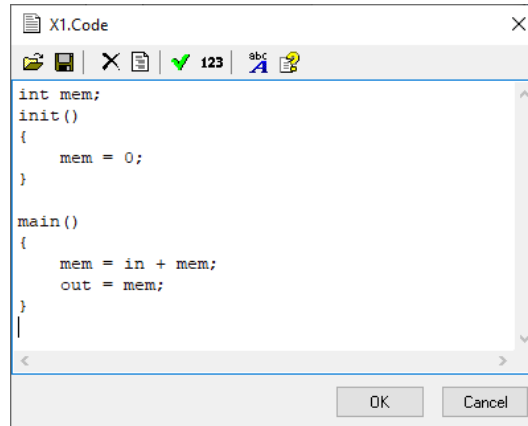
```
y1=1.2,y2=0,y3=2.345,integral=-4.19,counter=100
```

Here `y1`, `y2`, and `y3` are output variables; `integral` and `counter` are global variables.

IC string will be automatically modified by **Save initial conditions**  command: it fills in **IC** string with current variable values.

If not empty, **IC** will be executed right after initialization (`init` function), and before the first transient simulation step at $t=0$.

Editing C-code. To edit the **Code**, select the parameter and click  to open **Code** window:






```

int mem;
init()
{
    mem = 0;
}

main()
{
    mem = in + mem;
    out = mem;
}

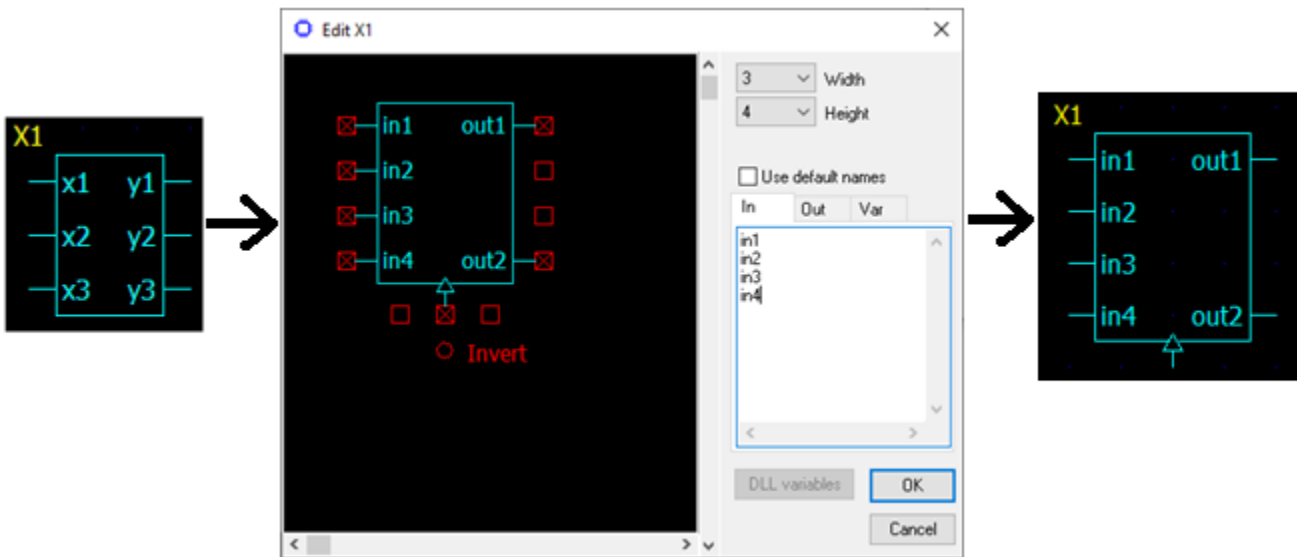
```

-  - clear code and set code template with empty `init()` and `main()` functions.
-  - check code. A message box with errors or warnings found will be shown.
- 123** - show line numbers (no editing allowed in this mode).
-  - Help (**F1**). Opens Help topic on statement, operator, or function, where cursor is located.

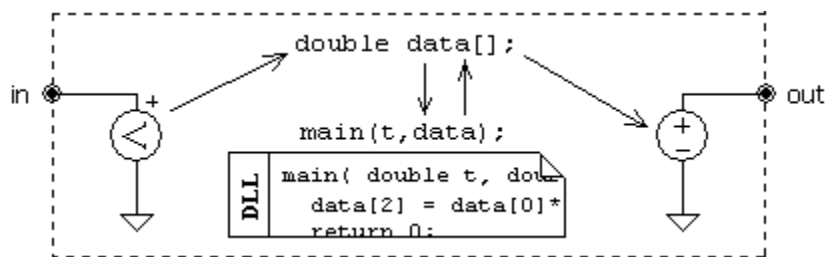
DLL component

In **DLL** component, the component function code can be written in C/C++ language, compiled by C compiler, and built as Windows DLL (Dynamic-Linked Library). DLL functions will be called by NL5 during transient simulation. DLL code will be executed much faster than C code in **C-code** component. However, changing the code requires recompiling the code and rebuilding the DLL.

DLL component is a customized component. Click **Edit component**  in the Components Window to open **Edit component** window, then specify symbol size, inputs, outputs, component variables, and clock (optional):



Execution. A principle of DLL code execution is shown on the following diagram:



A voltage on the input pin of the component **in** is measured by a “virtual” voltmeter and is assigned to the corresponding element of the array `data` of `double` type. The pointer to the array is passed as a parameter to the DLL function `main`. The function is executed; a new value of output **out** is evaluated and assigned to the corresponding element of the array `data`. A voltage equal to that value is set to a “virtual” grounded voltage source, connected to the corresponding output pin **out**.

Array `double data[]` is used to pass input/output values to/from DLL function. Array size is `<number of inputs> + <number of outputs>`. When calling DLL function, first `<number of inputs>` elements of the array are set to input values in the same order as input pins are shown on the component symbol (left

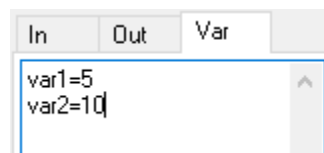
side, top-to-bottom). Output values calculated in the DLL are placed in the next *<number of outputs>* elements of the array in the same order as output pins are shown on the component symbol (right side, top-to-bottom). For example, if the component has two inputs and two outputs, the following code will assign minimum input value to the first output, and maximum input value to the second output:

```
data[2] = min(data[0], data[1]);
data[3] = max(data[0], data[1]);
```

Component variables. Any variables created in the DLL code are not accessible by NL5 lite. DLL **component variables** are used for storing values that are accessible both from DLL and NL5.

Component variables are instantiated in NL5 and can be accessed and modified both by NL5 and DLL. Component variables can be used, for example, for debugging purposes: they can be displayed on the graph as a trace without passing them through output pins of the component. Component variables can be changed by NL5 script or command line when simulation is paused. Also, values of component variables will be saved as initial conditions in the **IC** parameter of the component.

Input, output, and component variable names are defined in the In, Out, and Var tabs. You can also assign initial values (**IC**) to outputs and component variables in the form `name=value`:



If component variables are used, the size of the `data[]` array should be *<number of inputs> + <number of outputs> + <number of component variables>*. When calling DLL functions, corresponding elements of the array (following output values) are filled in with current values of component variables. DLL can modify component variables by assigning new values to corresponding elements of the array.

For example, if DLL component with 2 inputs and 2 outputs has a component variable `counter`, it can be modified by NL5 using following notation:

```
component_name.counter = 0;
```

DLL, in turn, can modify value of `counter` as:

```
data[4] = counter;
```

When simulation is already started and paused, open **Edit component** window for DLL component and click **DLL variables** button. DLL component variables and their current values will be shown (in floating point format). Those values can be edited and will be sent to the DLL when pressed **OK**.

Initialization function (Init parameter). The function should be declared as follows:

```
extern "C" __declspec (dlllexport) int NAME(double t, double* data);
```

where:

NAME – function name, for instance `init`
t – current simulation time (always = 0)
data – pointer to the array with input/output data

The function returns zero if no errors occur, or user-defined non-zero integer error code. If error occurs, the error code will be displayed in the NL5 error message window.

Initialization function is executed once at the beginning of simulation at $t=0$. The function can be used to assign initial (default) values to outputs and component variables (if defined) by setting corresponding elements of the array `data[]`, declare and initialize static DLL variables and arrays, allocate additional memory which will be used by DLL, open data logging files, etc.

This function is optional and can be omitted. Leave **Init** parameter blank if function is not used.

Main function (Main parameter). The function should be declared as follows:

```
extern "C" __declspec (dlllexport) int NAME(double t, double* data);
```

where:

NAME – function name, for instance `main`
t – current simulation time
data – pointer to the array with input/output data

The function returns zero if no errors occur, or user-defined non-zero integer error code. If error occurs, the error code will be displayed in the NL5 error message window.

The function calculates output variables using current values of input variables during transient simulation. If **clock** pin does not exist, the function is executed on every calculation step. If **clock** pin exists, the function is executed only on rising/falling edge of logical clock signal.

For example, function `main` below calculates sum of 8 inputs:

```
extern "C" __declspec (dlllexport) int main(double t, double* x)
{
    double y=0;
    for(int i=0; i<8; ++i) y += x[i];
    x[8] = y;
    return 0;
}
```

This function is optional and can be omitted. Leave **Main** parameter blank if function is not used.

Pause function (Pause parameter). The function should be declared as follows:

```
extern "C" __declspec (dlllexport) void NAME();
```

where:

NAME – function name, for instance `on_pause`

The function is called when transient simulation is paused. Use this function to perform operations such as saving data into the file, reallocate unused memory, etc.

This function is optional and can be omitted. Leave **Pause** parameter blank if function is not used.

Exit function (Exit parameter). The function should be declared as follows:

```
extern "C" __declspec (dlllexport) void NAME();
```


where:

NAME – function name, for instance `on_exit`

The function is called when DLL is being closed and DLL component destroyed. Use this function to free the memory allocated by other DLL functions, close files used for data logging, etc.

This function is optional and can be omitted. Leave **Exit** parameter blank if function is not used.


Using one DLL by several components. Several DLL components may use the same DLL file. Be aware that in this case only one copy of DLL is loaded into the memory, so that the same static variables will be shared by all components. In this case, **Init**, **Pause**, and **Exit** functions may be called only by one component. Also, different components can use different functions from the same DLL.

If you need several components to be using the same DLL with separate static variables, use several copies of the DLL file with different names. Another solution would be **attaching**  DLL to the component. If DLL is attached, NL5 will create a unique temporary DLL file for each component during simulation, so that each component will be using its own DLL copy.

Initial conditions (IC parameter). Initial conditions is a string with comma-separated expressions in “name=value” format assigning initial values to outputs and component variables of the DLL. For example:

```
min=-1.23,max=4.56,counter=100
```

where `min` and `max` are outputs, and `counter` is component variable.

IC string will be automatically modified by **Save initial conditions**  command: it fills in the string with current variable values.

If not empty, **IC** will be executed right before calling DLL initialization function, so that DLL initialization function could use parameters defined in the **IC** string.

Roots model

Roots model describes zeroes and poles of transfer function of **s-function** component:

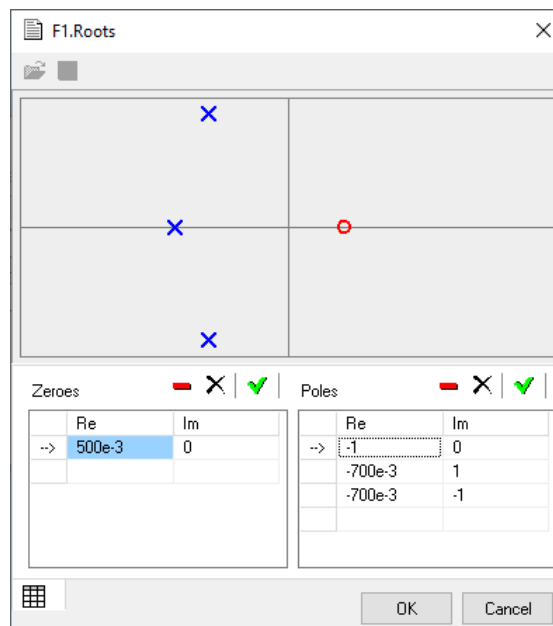
$$f(s) = \mathbf{K} * (s-z_1)*(s-z_2)... / (s-p_1)/(s-p_2)...$$

and impedance of **Impedance** component.

$$z(s) = \mathbf{Z} * (s-z_1)*(s-z_2)... / (s-p_1)/(s-p_2)...$$

where z_n are zeroes, p_n are poles.

Zeros and poles are entered in the **Roots** window:




To enter new zero/pole, select the last (empty) table cell, **Re** column, enter the value, and press **Enter**, or click to update the table. Then enter **Im** value if needed. If non-zero **Im** value is entered, a conjugate zero/pole is added automatically.

To edit existing zero/pole, select the cell, enter new value, and press **Enter**, or click to update the table. Click to remove selected zero/pole, marked by --> in the first column, or click to clear the table.

Attachments

Any files/data used by NL5 components (**File**, **SubCir**, **DLL**, etc.) can be **attached** to the NL5 schematic file. Then the schematic file will contain all the data needed for simulation, which makes it easy to maintain and distribute.

To attach the data to the component, click **Attach**  on the **Component window** toolbar. To remove attachment click **Remove attachment** .

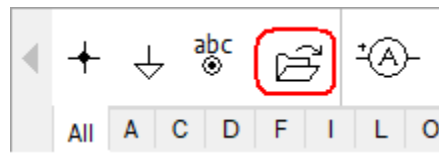
Please note that attaching **DLL** may result in large size of the schematic file, since binary data of the dll file are converted into text format.

NL5 components (*.nlc)


NL5 component can be saved into a **component file**: schematic file with `nlc` extension. Then the component can be loaded from the file and placed onto schematic as a standard component. For components with external data (**SubCir** and **File** models, **DLL**), attach the data to the component and save it in the component file: then the **component file** can be easily distributed without additional files.

NL5 lite cannot save component into component file.

To place component onto schematic, click **Load component** in the **Components bar**:



Customized component

Some components are **customized**, so that their symbol and/or number of pins can be edited by user. To edit component symbol, click **Edit component**  on the **Components window** tool to open **Edit component** window.

The following component properties can be edited for all customized components:

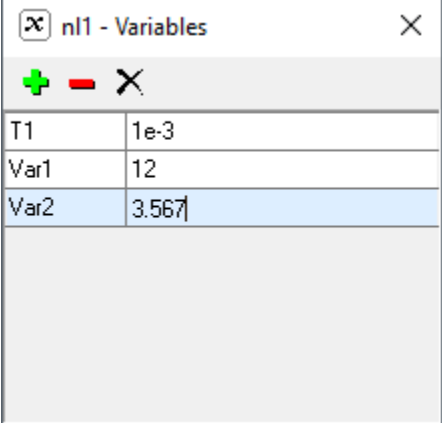
- Symbol width.
- Symbol height.
- Number of pins (inputs and/or outputs).
- Pins location.

The following properties can be edited for specific component types:

- Pin names.
- Subcircuit label names.
- Clock signal, inversion of clock signal.
- Inversion of the output.
- Logical function.
- Number of windings.
- Beginning and end of windings.
- DLL component variables.

Schematic variables

Schematic variable is a floating point variable which belongs to the schematic. The variable and its value can be defined in the **Variables window**:





The screenshot shows a window titled "nl1 - Variables" with a close button (X) in the top right corner. Below the title bar is a toolbar with three icons: a green plus sign, a red minus sign, and a black X. The main area contains a table with three rows. The first row has "T1" in the left column and "1e-3" in the right column. The second row has "Var1" in the left column and "12" in the right column. The third row has "Var2" in the left column and "3.567" in the right column. The third row is highlighted with a light blue background.

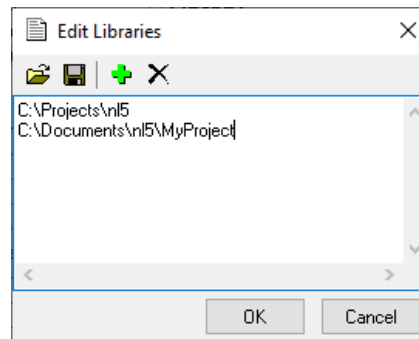
T1	1e-3
Var1	12
Var2	3.567


Variables can be used in the expressions, for example when defining component parameters.

Schematic settings

Details on some **Schematic settings**  tabs:

Library. Specify library paths list for component files (**DLL**, **SubCir**, **File** model, etc.). Click **Edit** , then enter library paths manually, one path per line, or click  to select from folders list:




When new schematic is created, a default list is copied from **Preferences** , **Library** tab. Library path could be full path, or path relative to the directory where schematic file is located.


If specified file name of the component does not have a full path, NL5 will search for the file in the following order:

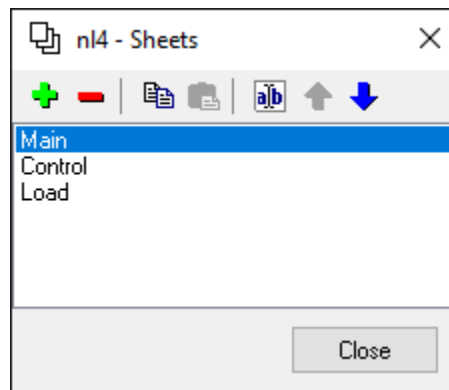
- In the directory where schematic file is located.
- In the directories specified in the **Library** list (see **Schematic settings** chapter).
- In the directories specified in the **Library** list relative to the directory of schematic file.

Components. Specify logical levels and threshold for all components in the schematic.


Schematic sheets

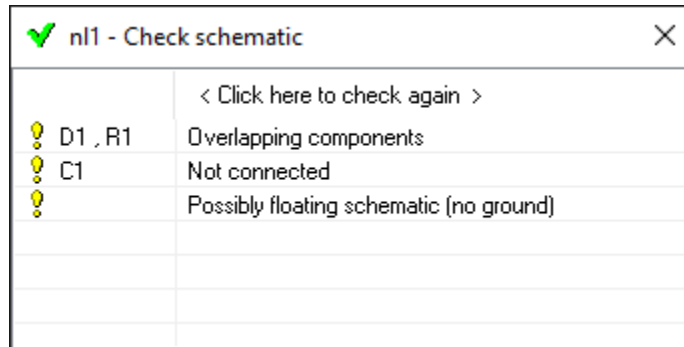
Schematic may contain several **Sheets** . Electrical connection between schematic sheets can be done through labels and functions.

Existing sheets are shown in the **Sheet tab** of **Schematic window**. **Right-click** on the tab to access sheets-related commands in the pop-up context menu, or click **Sheets**  to manage sheets in the **Sheets** window:




Check schematic

Check schematic  command performs extensive check of the schematic for potential problems, and reports results in the window:



Also, the check function removes unnecessary connection points and adds connection points where needed (three wires).

Click on the row to select (highlight) a schematic element(s) for which the problem was detected.

Go to **Preferences**  , **Warnings** tab to select/unselect reported issues.

Right-click to access check-related commands in the pop-up context menu.

Schematic check is automatically performed at **Transient** and **AC** simulation start. The problems reported by the function do not prevent from running simulation, and typically do not affect simulation results. However, they could indicate not accurate schematic design, and should be fixed if possible.

IV. Transient

Algorithm

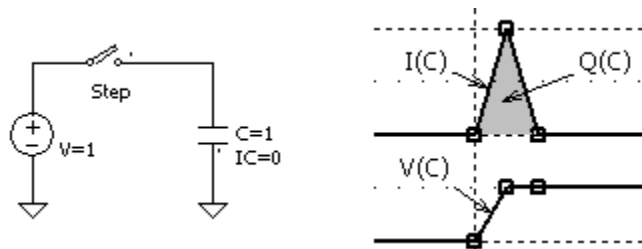
NL5 lite and NL5 perform simulation with ideal and piecewise linear (PWL) components. For instance, an ideal diode is either short circuit (zero resistance), or open circuit (infinite resistance), so that its PWL representation consists of just two linear segments. As long as all of the components are staying within their current segment, the circuit is described by the system of linear differential equations. The system is modified only at the moments when at least one component changes its linear segment.

Start. When simulation starts at $t=0$, a **Direct Current (DC) operating point** is calculated first. Some of the components may have specified Initial Condition (IC): these ICs will be used during DC operating point calculation. If circuit has more than one steady state, it can be set to a desired state by defining proper ICs. After DC operating point is found, simulation continues as a sequence of **linear segment** simulations, with **switching** between them.

Linear segment. In the linear segment, the circuit is described by the system of linear differential equations, which is solved by Trapezoidal integration method. The method provides sufficient accuracy with good robustness and calculation speed. During linear segment simulation, the algorithm is performing “switching point detection”: checking conditions on all components that may change their state or linear segment.

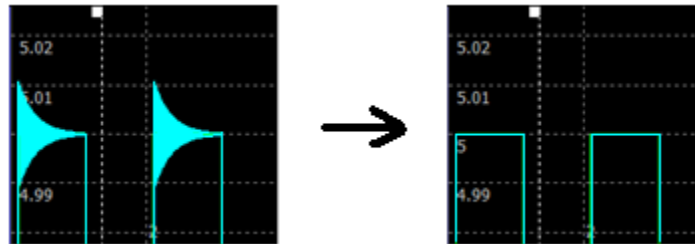
Switching. When switching point is detected, the current linear range ends, **switching** algorithm performs instantaneous switching, and a new linear segment starts. Switching algorithm can be optimized for specific circuit by selecting from several options at **Transient settings/Advanced settings**.

Instantaneous switching of ideal components may produce infinite voltage or current pulses. For example, when capacitor is connected to voltage source through ideal switch, an infinitely short current pulse with infinite amplitude may occur. The area (integral over time) of such pulse is limited and is equal to the total charge coming to or out of the capacitor during switching. Similar situation may occur when current through the inductor is discontinued, which results in an infinite voltage pulse across the inductor. Integral of the voltage over time corresponds to a magnetic flux in the inductor. Such a pulse is known as Dirac pulse, or delta-function. In NL5, the current or voltage delta-function is shown as a triangle pulse with the duration of each slope equal to minimal calculation step used at that moment, and the area satisfies charge or magnetic flux conservation law.



Simulation step. Unlike many analog simulators, NL5 does not perform automatic step control. Selecting simulation step is user's responsibility. This gives user full control on simulation, although requires certain experience and understanding of the process. The rule of thumb is keeping simulation step below smallest time constant in the circuit, otherwise the integration method may get unstable, and produce “numerical oscillations” when signal is “jumping” up and down at each simulation step.

Very often, such oscillation caused by non-optimal step would not affect overall functionality of the circuit and can be ignored. To get rid of that type of oscillation, reduce simulation step to reasonably small value, or use **Suppress oscillations** option (**Transient settings/Advanced settings/Transient/Simulation step**). Here is an example of transient improvement with this option:



Although simulation step is specified by user, NL5 still can **automatically reduce the step** to satisfy the following conditions:

- Period of sine source contains at least 16 steps.
- Pulse or switch On/Off state contains at least 4 steps.
- Non-zero rising or falling edge contains at least 4 steps.
- Interval between two points of the source component, interpolating the signal between two points, contains at least 4 steps.
- Delay time of transmission line and “delay” component contains at least 2 steps.

Also, simulation step can be reduced to detect switching point and perform accurate switching.

Automatic step reduction can be disabled by selecting **Constant step** option (**Transient settings/Advanced settings/Transient/Simulation step**).

Traces. During simulation, NL5 only stores data for specified **traces**. Pause or stop simulation to add/remove traces, then continue simulation or start simulation again.

Memory. Simulation data is stored in the operating memory. The memory is allocated as needed by relatively small blocks. If available operating memory is not enough for all the data, some of the blocks currently storing the very beginning of the trace will be reassigned to a new data, and thus some traces may be “truncated” at the beginning. At the first time this happens, the warning message will be displayed in the status bar of **Transient window**.

By default, NL5 saves all calculated data for all traces. To reduce memory consumption, a simple data compression method can be used: if several data points in a row have the same amplitude, only first and last point of such sequence are being stored. To use this method, select **Data compression** option (**Transient settings/Advanced settings/Transient**).

Running transient



- Go to **Settings window** (select desired Transient settings tab).



- **Start** transient (available on the **Schematic** and **Transient window** toolbars). When simulation is running, schematic cannot be edited, and component parameters/model cannot be changed. **Stop** or **Pause** simulation to do any changes.




- **Pause** transient (or press **Space**).



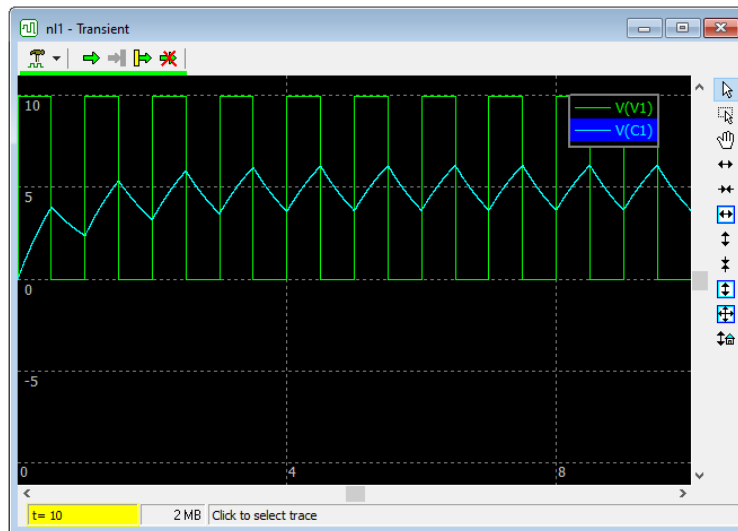
- **Continue** transient (or press **Space**).



- **Stop** transient. Clear extra memory allocated for simulation. Simulation cannot be continued.



Simulation is always paused at the right edge of the screen. Please note that simulation step is not adjusted to stop exactly at the screen edge, so the last calculated point could have time greater than screen edge. Press **Space** or click **Continue**  to continue simulation.



Transient window



To add transient traces, go to **Add traces**  tab of **Settings window** .

Go to **Format traces**  tab of **Settings window**  to select trace type, color, scales, etc.


Go to **Screen**  tab of **Settings window**  to configure **Transient window** scales (horizontal only).


Vertical scale. Each trace is shown with its individual **Scale** and **Mid** values, specified at **Format traces**  tab of **Settings window** . Vertical scale and gridlines are shown for selected trace, or several selected traces if their **Scale** and **Mid** values are the same. If several traces with different **Scale** and **Mid** values are selected, vertical scale is not shown.

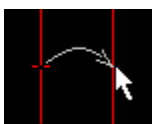
All vertical zoom/scroll/reset operations (buttons, keys, mouse, mouse wheel) will change scales and offsets of all traces accordingly.

Click **Reset vertical scale**  to set **Scale** and **Mid** of all traces to default values.

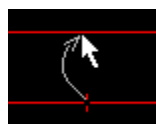
Mouse. There are 3 modes of **mouse operation** in analog sections:

Cursors  mode. In this mode, **double-click** to center the screen to the mouse pointer position.

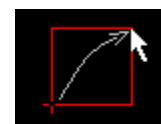
Zoom  mode (press and hold **Ctrl** key to switch to **zoom** mode temporarily). Click and drag to select horizontal, vertical, or rectangle area to be zoomed-in:




Move mouse pointer horizontally to select horizontal area.



Move mouse pointer vertically to select vertical area.



Move mouse pointer diagonally to select rectangle area.

Scrolling  mode (press and hold **Shift** key to switch to **scrolling** mode temporarily). Click and drag to scroll the screen.

Mouse wheel:


Mouse wheel – horizontal zoom (relative to mouse pointer position).


Shift-mouse wheel – horizontal scroll.


Ctrl-mouse wheel – vertical zoom (relative to mouse pointer position).

Ctrl-Shift-mouse wheel – vertical scroll.

Keys and shortcuts:

F5 – show transient window .

F6 - start transient .

F7 - stop transient .

Space – pause/continue transient.

Up, Down – vertical scroll.

PgUp, PgDn – vertical zoom , .



Home – vertical fit the screen, same as .


Left, Right – horizontal scroll.

End – center beginning of traces (set to the middle of the screen).


Ctrl-End – center end of traces.

Shift-End – center middle of the traces.


Ctrl-PgUp, Ctrl-PgDn – horizontal zoom , .


Ctrl-Home – horizontal fit the screen .

Shift-PgUp, Shift-PgDn – vertical and horizontal zoom.

Shift-Home – fit the screen .

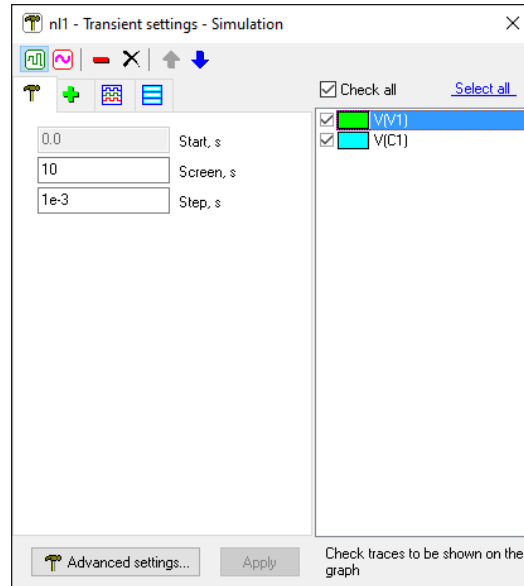
Status bar shows current transient time, simulation status (running, paused), and amount of memory allocated for calculation and data.

Legend  shows list of traces currently displayed on the graph.

- **Click** on the border of the legend and drag to move the legend window.
- **Click** on the trace in the legend to select one trace.
- **Ctrl-click** to select more than one trace or un-select the trace.
- Go to **Preferences** , **Legend** tab to select legend style (colors, font).

Transient settings

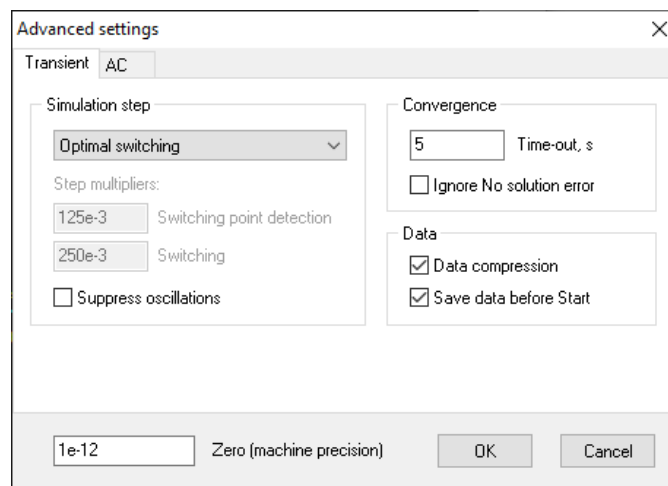
Simulation



Screen. Screen size settings at simulation start (simulation always starts at $t=0$).

Step. Maximum calculation step. Actual step may be reduced by the algorithm if needed.

Advanced settings  - more algorithm, transient, and data settings.



Simulation step. Select simulation step control mode:

- **Constant step** – step will never be reduced by NL5 and will always be as defined by the user. This option provides fastest simulation, however switching and convergence may not be reliable.
- **Fast switching** - fast simulation with acceptable switching convergence for most circuits.
- **Optimal switching** - good convergence and performance for most circuits.
- **Accurate switching** - best convergence, however simulation time might be longer.
- **Custom** – use your own **Step multipliers**:

Switching point detection - current step may be reduced by this coefficient to detect switching point more accurately.

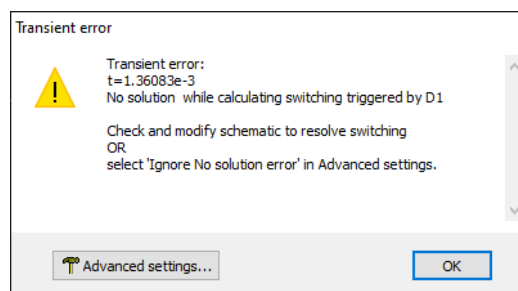
Switching – after switching occurred, current step will be reduced by this coefficient for at least two simulation steps to provide more accurate results. Also, this will define the displayed amplitude of possible “infinite” voltage or current pulses for simulation with ideal components. This value is recommended to be at least two times greater than **Switching point detection** value.

Suppress oscillations. Reduce oscillations caused by high simulation step. Using this option may increase simulation time.

Convergence.

Time-out specifies maximum time allowed for calculating one simulation step. If calculation takes longer, simulation is stopped with error message. If time-out occurred due to unresolved switching iterations, the error message will indicate a component that started switching process. Time-out may also occur due to infinite while/do/for loops of C-code. Set **Time-out = 0** to disable time-out detection.

Ignore No solution error. Sometimes switching process cannot be resolved, and error message is being displayed, for example:



Such a problem is most likely to happen during switching process in a complex circuit with ideal components, where infinite loop gain, infinite voltage/current, or other extreme (non-realistic) conditions may occur. To resolve the problem:

- First, check the schematic to fix possible general schematic issues.
- Identify critical ideal components which may cause the problem and replace them with more “realistic” models (use finite gain and bandwidth, non-zero and non-infinite resistance, etc.).

- Add small resistors in series, large resistors in parallel, and/or small capacitors in parallel to critical points.

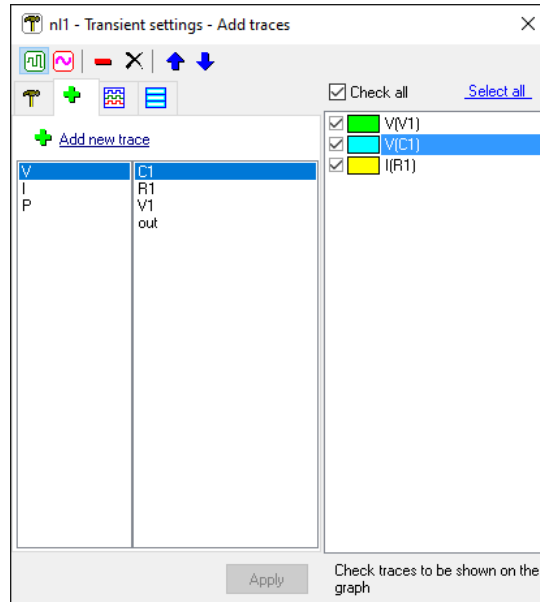
If none of those actions help, select **Ignore 'No solution' error** option. With this option enabled, if switching cannot be resolved, NL5 will accept some, probably invalid, solution, and continue transient simulation. Although one or more data points may not be entirely correct, there is a high chance that the solution will converge to the correct one very soon.

Data.

Data compression. If several data points in a row have the same amplitude, only first and last point of such sequence will be stored.

Zero (machine precision) – relative difference between two floating point numbers which can be reliably recognized. Increasing **Zero** may help to improve convergence at switching points, however, for most of circuits changing **Zero** is not recommended.

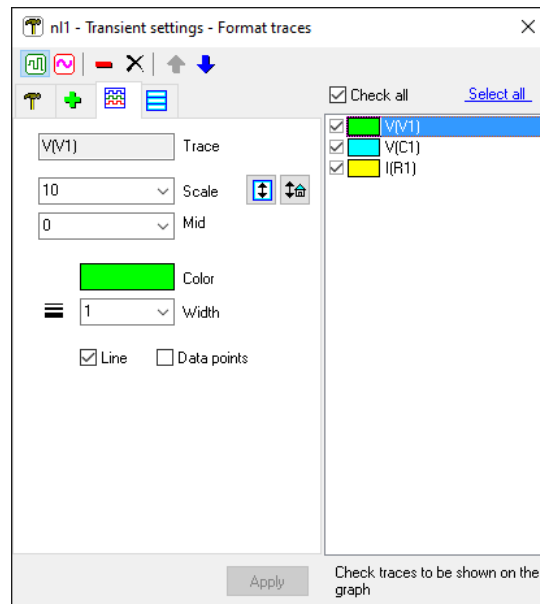
Add traces



Select trace type to add: **V, I, P.**

Select one or more components in the list, then click **Add new trace**, or **double-click** on the component in the list.

Format traces




Check traces to be shown on the transient graph.



Select one or more traces (using **Ctrl** and **Shift** keys) to perform operations on selected traces.

Click **Select All** to select all traces. Traces can also be selected in the **Legend** area of the Transient window.

The top toolbar provides various operations on selected traces and trace data.

If more than one trace of different type selected, only parameters applicable to **all** selected traces will be shown. If parameter has different values for different traces, its field will be blank. If color is different, it

will be displayed like this: . If a new value is entered in an empty field, that value will be assigned to all selected traces.

Scale, Mid – scale values for analog trace. Click  to select from previously used values. Click  to auto-scale, or  to reset vertical scale of selected traces.

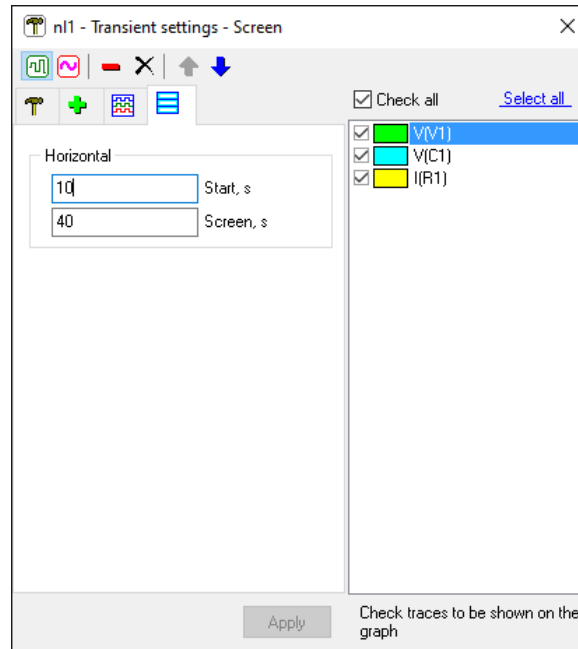
Color - click colored rectangle to select trace color.

Width - enter or select from drop-down list trace width in pixels.

Line – show lines.

Data points – show simulation data points. If **Data compression** option is selected, some simulation data points of a constant signal will not be stored and displayed.

Screen



View and modify screen horizontal scale.


V. AC analysis

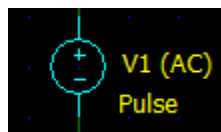
Algorithm

There is one methods of AC analysis in NL5 lite: **Linearize schematic**.

Linearize schematic is a standard low-signal AC analysis. First, all non-linear components are replaced with linear equivalents at their operating point. Second, a signal of specified frequency with unit amplitude and zero phase is applied to the input node, and signals at other nodes are found by solving a system of linear equations. The process is repeated for specified number of frequencies.

In order to linearize schematic, states of all components should be known. It can be done manually by setting Initial Conditions (IC) for non-linear components, diodes, and controlled switches, or by automatic calculation of DC operating point (**Calculate DC operating point** check box in the **AC Settings**). DC operating point is calculated exactly as in transient analysis.

AC source . Any voltage source, current source, or label can be used as AC source. Component selected as AC source will be marked with “(AC)” text on the schematic and in the **Components window**:



R2	1
R4	1
V1 (AC)	Pulse

The AC source component may have any model (except **SubCir**, **Label**, and **IC**): the model will be ignored for **Linearized schematic** method. DC voltage/current of the component will be set to its DC value at $t=0$, and AC voltage/current required for AC analysis will be added to that DC level.

Running AC analysis



- Go to **Settings window** (select desired AC settings tab).

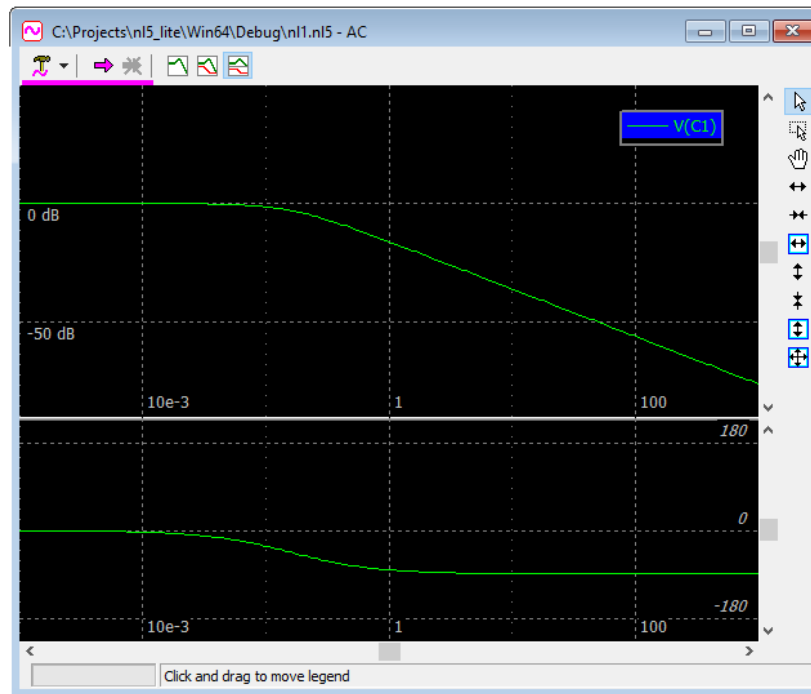








- **Start AC** (available on the **Schematic** and **AC window** toolbars). When simulation is running, schematic cannot be edited, and component parameters/model cannot be changed.



- **Stop AC**.




AC window



To add AC traces and specify AC source, go to **Add traces**  tab of **Settings window** . Go to **Format traces**  tab of **Settings window**  to select trace color, scales, etc. Go to **Screen**  tab of **Settings window**  to configure **AC window** scales.


Scales. All traces are shown with the same scales, which can be linear or logarithmic.

Phase display method selection:

- **Phase off**  – do not show phase.
- **Phase On**  - show phase in the same window with other traces.
- **Phase separate**  - show phase in the select **Phase** section.

Mouse. There are 3 modes of **mouse operation**:

Cursors  mode: **double-click** to center the screen to mouse pointer position (magnitude only).

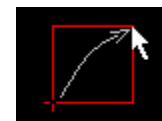
Zoom  mode (press and hold **Ctrl** key to switch to **zoom** mode temporarily). Click and drag to select horizontal, vertical, or rectangle area to be zoomed-in:




Move mouse pointer horizontally to select horizontal area.



Move mouse pointer vertically to select vertical area.



Move mouse pointer diagonally to select rectangle area.

Scrolling  mode (press and hold **Shift** key to switch to **scrolling** mode temporarily). Click and drag to scroll the screen.

Mouse wheel:

Mouse wheel – horizontal zoom (relative to mouse pointer position).

Shift-mouse wheel – horizontal scroll.

Ctrl-mouse wheel – vertical zoom (relative to mouse pointer position).

Ctrl-Shift-mouse wheel – vertical scroll.

Keys and shortcuts:

F8 – show AC window .

F9 - start AC .

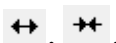
Tab – toggle **Phase** display mode .


Up, Down – vertical scroll.

PgUp, PgDn – vertical zoom .


Home – vertical fit the screen .

Left, Right – horizontal scroll.


Ctrl-PgUp, Ctrl-PgDn – horizontal zoom .


Ctrl-Home – horizontal fit the screen .

Shift-PgUp, Shift-PgDn – vertical and horizontal zoom.

Shift-Home – fit the screen .

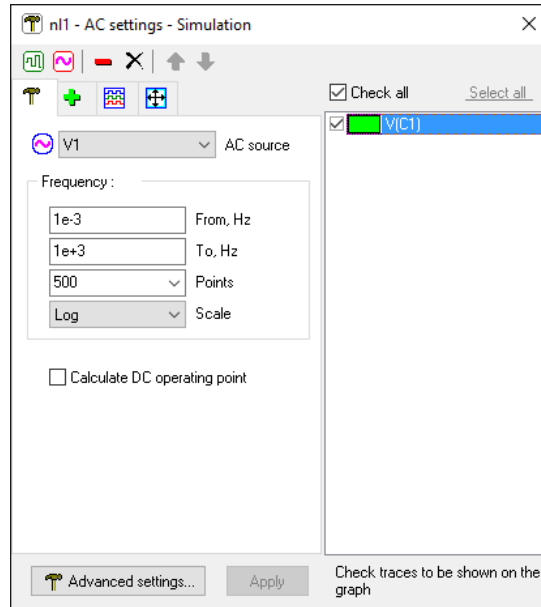
Status bar shows current frequency of AC simulation (progress).

Legend  shows list of traces currently displayed on the graph.

- **Click** on the border of the legend and drag to move the legend window.
- **Click** on the trace in the legend to select one trace.
- **Ctrl-click** to select more than one trace or un-select the trace.
- Go to **Preferences** , **Legend** tab to select legend style (colors, font).

AC settings

Simulation



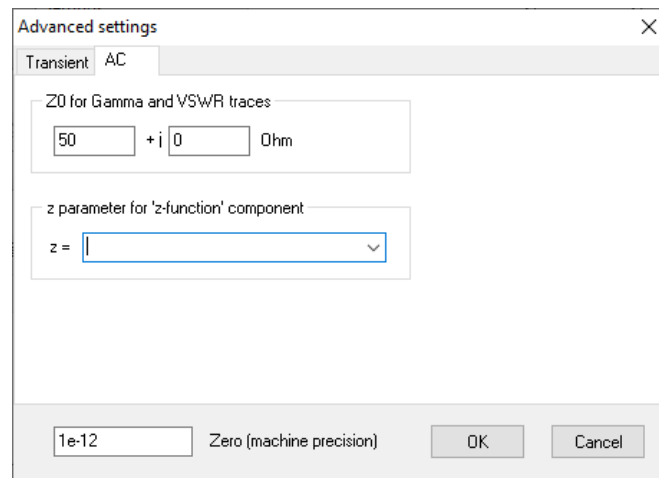
AC source – enter the name of AC source component or select the name from drop-down list.

Component selected as AC source will be marked with **(AC)** text on the schematic and in the **Components window**.

Frequency - select frequency, number of points, and scale type.

Calculate DC operating point – use this option for non-linear schematic if needed.

Advanced settings  - more AC settings.

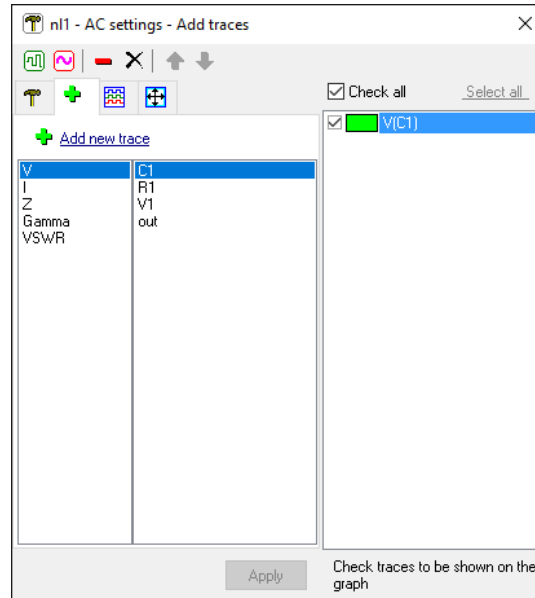


- **Z0 for Gamma and VSWR** - characteristic impedance for Gamma and VSWR traces.
- **Z parameter** - define custom formula for Laplace-space approximation of z-transform parameter z or select formula from drop-down list. For example:

```
exp (s*1e-6)
exp (s*T)
(2+s*T) / (2-s*T)
```

This parameter is used only by **z-function** components.

Add traces



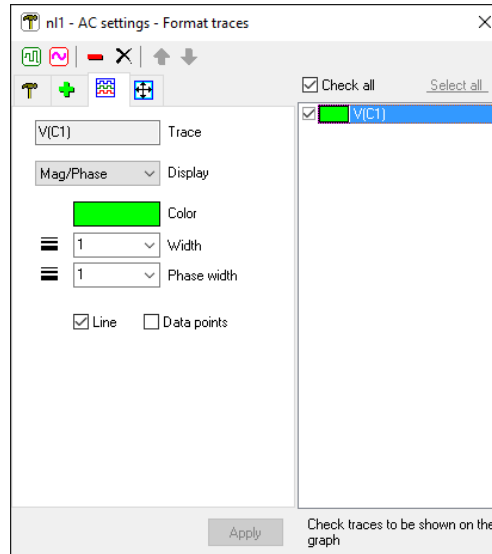
V, I:

Select one or more components in the list, then click **Add new trace** or **double-click** on the component in the list.

Z, Gamma, VSWR:

Click **Add new trace** or **double-click** on the trace type name in the list.
 Show impedance (**Z**), reflection coefficient (**Gamma**), or voltage standing-wave ratio (**VSWR**) relative to AC source.

Format traces




Check traces to be shown on the AC graph.

Select one or more traces (using **Ctrl** and **Shift** keys) to perform operations on selected traces. Click **Select All** to select all traces. Traces can also be selected in the **Legend** area of the AC window.

The top toolbar provides various operations on selected traces and trace data.

If more than one trace of different type selected, only parameters applicable to **all** selected traces will be shown. If parameter has different values for different traces, its field will be blank. If color is different, it

will be displayed like this:  **Color**. If a new value is entered in an empty field, that value will be assigned to all selected traces.

Display defines how the trace will be displayed:

- **Mag/Phase** - magnitude and phase.
- **Re** – real part only.
- **Im** – imaginary part only.

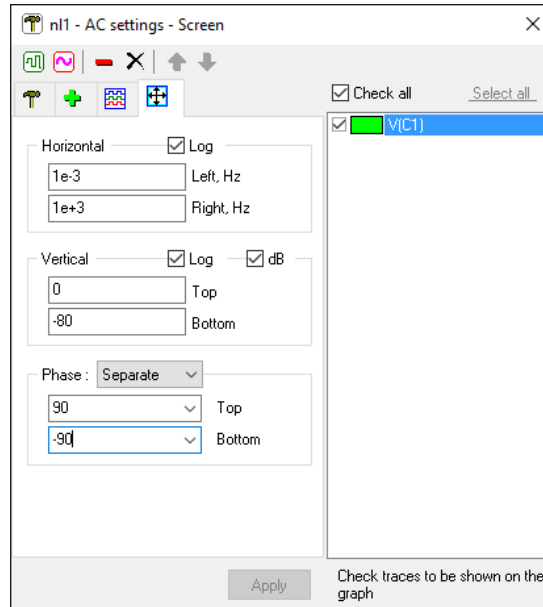
Color - click colored rectangle to select trace color.

Width - enter or select from drop-down list trace width in pixels.

Line – show lines.

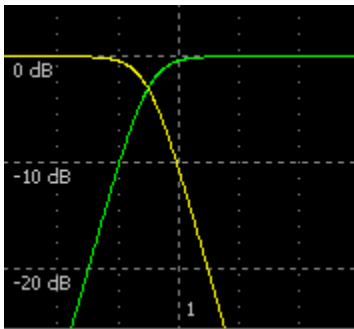
Data points – show simulation data points.

Screen

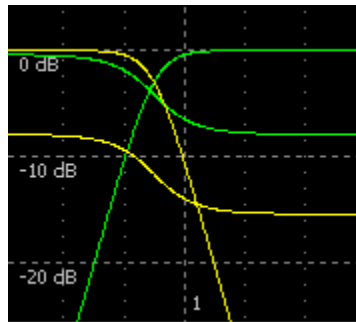


Phase - select phase display mode:

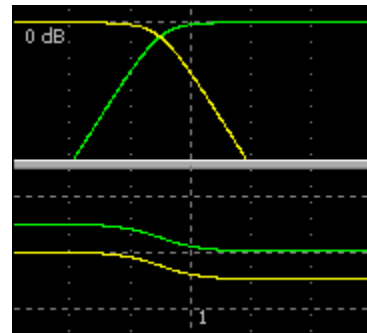
- **Off** - do not show phase.
- **On** - show magnitude and phase in the same area of the graph.
- **Separate** - show magnitude and phase in separate areas of the graph.



Phase Off



Phase On



Phase separate

Press **Tab** in the AC Window to toggle **Phase** display mode.