NATOMAS ENGINEERING

natomaseng.com

SimPad

Simulations Notepad

User Guide

01.09.2024

Table of contents

1. General information

1.1 Purpose

1.1.1 **SimPad** – a Simulations Notepad, is a text editor with macro processing and Python scripting to prepare input data for simulations solvers and perform calculations and result data analysis

1.2 The main functions

1.2.1 Creation and editing of text files of source data for calculation programs.

1.2.2 Calculation of text files based on the specified data and parameters.

1.2.3 Cloning and duplication function, which allows you to create copies of files based on a template, automatically replacing variables with the desired values.

1.2.4 Various options for viewing calculation results, including tables for detailed data analysis, graphs for visualizing results and reports for obtaining information in a text format with the ability to print or save.

2. Terms and definitions

2.1 Basic terms

2.1.1 **Processed files (SimPad project files)** are text files containing macro processor data blocks.

2.1.2 **Handler** – a batch file (bat file) or a group of batch files that receive the name (path) of the working folder, the name (path) of the file with post-processor parameters, the names (paths) of the processed files, handler parameters, and calling programs (including a script or powershell scripts) to process files in the desired sequence.

2.1.3 **Viewer** is a batch file (bat file) that receives the name (path) of the working session folder in the form of startup parameters (keys), and calls the program to view the results of processing.

2.1.4 **Session** is a concept that advises starting the procedure for processing the resulting macro substitution of the source file. Sessions are indicated by an ordinal number. Each session corresponds to a folder with a name in the form of a session number, located in the workspace (folder) of the program specified in the configuration file.

2.1.5 **Calculation session** – a session for launching a handler for a separate calculation with one set of files obtained for one set of macro-variable values.

2.1.6 **Batch session** – a session in which several calculation sessions are performed, after which the results can also be processed.

2.1.7 **Data blocks** are a group of consecutive rows, starting from the row with the block identifier to the row with the identifier of the next block.

2.1.8 **A block identifier** is a unique name used to identify and distinguish blocks of data during processing.

2.1.9 **Syntax highlighting** is a text editor function that automatically highlights different elements of code or text in different colors or fonts according to their syntactic role. The highlighting is set in the syntax file.

2.1.10 **Syntax highlighting style (syntax file)** is a text file with lines for specifying regular expressions of syntax blocks and font parameters.

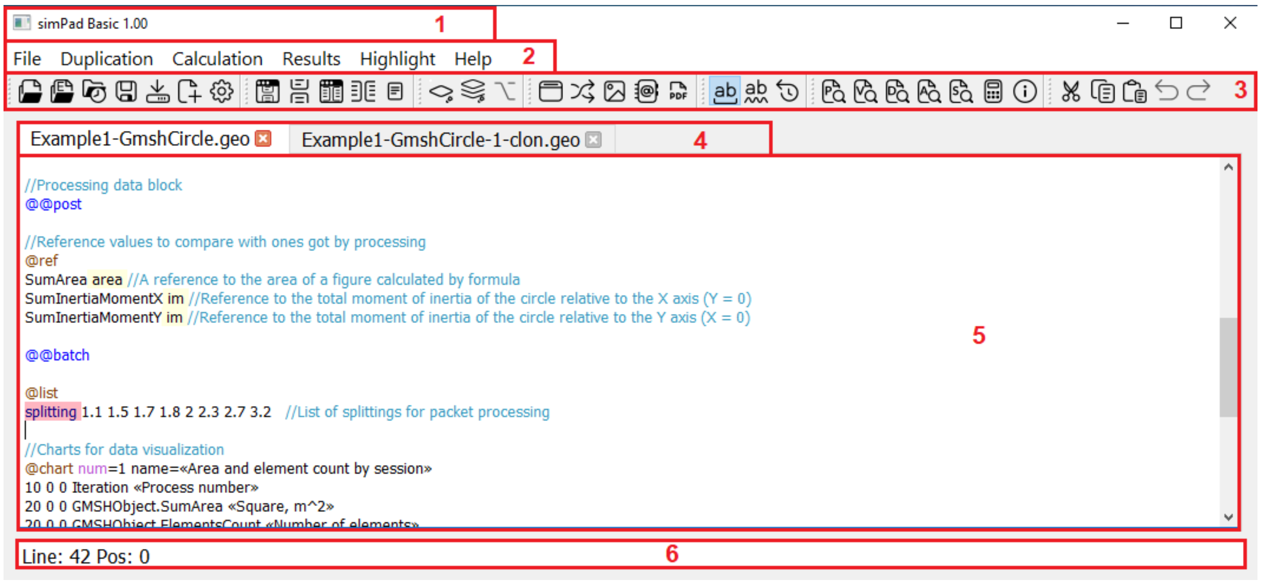
2.1.11 **A regular expression** is a template for searching and processing text that allows you to find strings that match a given pattern.

2.1.12 **A macro processor** is a component of a program that automatically processes text files by performing macro substitutions.

2.1.13 P**ostprocessor** – a program or script that interprets the results of the handler and converts them into a format readable by simPad (_property file).

3. Working with the program

3.1.1 The main window :


simPad Basic 1.00
1
File Duplication Calculation Results Highlight Help
2
3
Example1-GmshCircle.geo
Example1-GmshCircle-1-clon.geo
4
5
6

```
//Processing data block
@@post

//Reference values to compare with ones got by processing
@ref
SumArea area //A reference to the area of a figure calculated by formula
SumInertiaMomentX im //Reference to the total moment of inertia of the circle relative to the X axis (Y = 0)
SumInertiaMomentY im //Reference to the total moment of inertia of the circle relative to the Y axis (X = 0)

@@batch

@list
splitting 1.1 1.5 1.7 1.8 2 2.3 2.7 3.2   //List of splittings for packet processing

//Charts for data visualization
@chart num=1 name=«Area and element count by session»
10 0 0 Iteration «Process number»
20 0 0 GMSHObject.SumArea «Square, m^2»
20 0 0 GMSHObject.ElementsCount «Number of elements»
```

```
Line: 42 Pos: 0
```

3.1.2

3.1.3 **(1) The title line** is the name of the program, version.

3.1.4 (**2) Menu** – the program menu.

3.1.5 **(3) Toolbar** – contains buttons for frequently used commands.

3.1.6 **(4) Open Files panel** – displays tabs of open, source files with names, as well as output files. The active file displayed in the edit window highlights the file tab in white on the file panel. The output files have a postscript in their name in the form of a number and the inscription clone or param, depending on the selected type of processing of the source file. The extension for the output file can be set in the @datafile block, in the ext parameter.

3.1.7 **(5) Edit field** – the text file editing field.

3.1.8 **(6) Cursor field** – the field for displaying the cursor location in the text of the edited file.

3.2 File Menu

3.2.1 **New File** – Creates a new file and gives you a choice of a set of templates with initially specified data blocks.

3.2.2 **Open File** – Opens the file in the simPad editing window, initializes syntax highlighting according to the specified file extension.

3.2.3 **Example** – Opens a file from the Sample folder and copies it to the SITIS\Prj\simpad directory for further editing.

3.2.4 **Last File** – Opening the last file.

3.2.5 **Recent Files** – A list of the most recently opened files.

3.2.6 **Save** – saves changes to the current file.

3.2.7 **Save As** – opens the file saving dialog.

3.2.8 **Settings** – opens the settings window where you can change the language of the program, and the font size of the text and menu.

3.2.9 **Exit** – closing the program.

3.3 Duplication Menu

3.3.1 **Duplicate ...** – opens the macro variable editing window for further duplication.

3.3.2 **Duplicate** – duplicate the current file without opening the macro variable window.

3.3.3 **Clone ...** - opens the macro variable editing window for further cloning.

3.3.4 **Clone** – clone the current file without opening the macro variables window.

3.3.5 **Show** – open the last duplicated/cloned file.

3.4 Calculation Menu

3.4.1 **Single Calculation** – calculation of the file by the selected processor.

3.4.2 **Packet Calculation** – batch processing of the file by the selected processor.

3.5 The Results menu

3.5.1 **Properties** – Displays a table of properties and macro variables for the current file.

3.5.2 **Graphic** – Opens the graph selection window for the current file.

3.5.3 **Image** – Opens the image display window.

3.5.4 **Viewers** – Opens the viewer selection window.

3.5.5 **Reports** – Opens the report display window.

3.6 Highlight Menu

3.6.1 **Static** – Static syntaxis highlight switch.

3.6.2 **Dynamic** – Dynamic syntaxis highlight switch.

3.7 Help menu

3.7.1 **Processors Info** – Information about solvers and processors.

3.7.2 **Viewers info** – information about viewers.

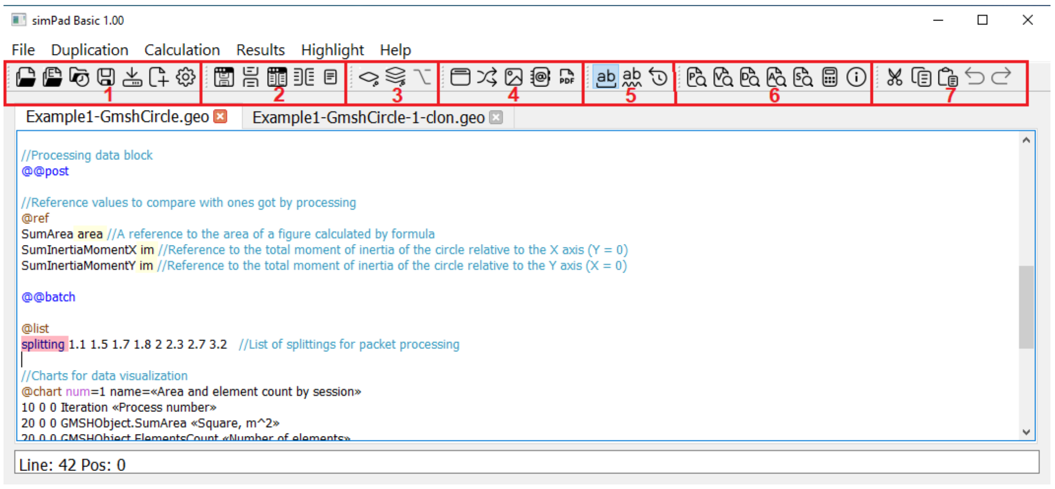
3.7.3 **API** – opening the API documentation.

3.7.4 **Round** – opens the rounding numbers window.

3.7.5 **Highlight Check** – Checking the style file for validity.

3.8 The Toolbar

3.8.1 The toolbar is divided into seven groups, which combine the tools from the corresponding menu items.


simPad Basic 1.00
File Duplication Calculation Results Highlight Help
1
2
3
4
5
6
7
Example1-GmshCircle.geo
Example1-GmshCircle-1-clon.geo
Line: 42 Pos: 0

```
//Processing data block
@@post

//Reference values to compare with ones got by processing
@ref
SumArea area //A reference to the area of a figure calculated by formula
SumInertiaMomentX im //Reference to the total moment of inertia of the circle relative to the X axis (Y = 0)
SumInertiaMomentY im //Reference to the total moment of inertia of the circle relative to the Y axis (X = 0)

@@batch

@list
splitting 1.1 1.5 1.7 1.8 2 2.3 2.7 3.2   //List of splittings for packet processing

//Charts for data visualization
@chart num=1 name=«Area and element count by session»
10 0 0 Iteration «Process number»
20 0 0 GMSHObject.SumArea «Square, m^2»
20 0 0 GMSHObject.ElementsCount «Number of elements»
```

3.8.2

3.8.3 **(1)** Tools from the File menu item.

3.8.4 **(2)** Tools from the Duplication menu item.

3.8.5 **(3)** Tools from the Calculation menu item.

3.8.6 **(4)** Tools from the Results menu item.

3.8.7 **(5)** Tools from the Highlight menu item.

3.8.8 **(6)** Tools from the Help menu item.

3.8.9 **(7)** Text editing tools.

3.8.9.1 **The Cut selected text** button cuts the selected text.

3.8.9.2 **The Copy selected text** button copies the selected text.

3.8.9.3 **Paste text from clipboard** button– inserts previously cut or copied text.

3.8.9.4 The button cancels the user's last action during text editing.

3.8.9.5 Redo previous Undo action button – repeats the action that was canceled using the button .

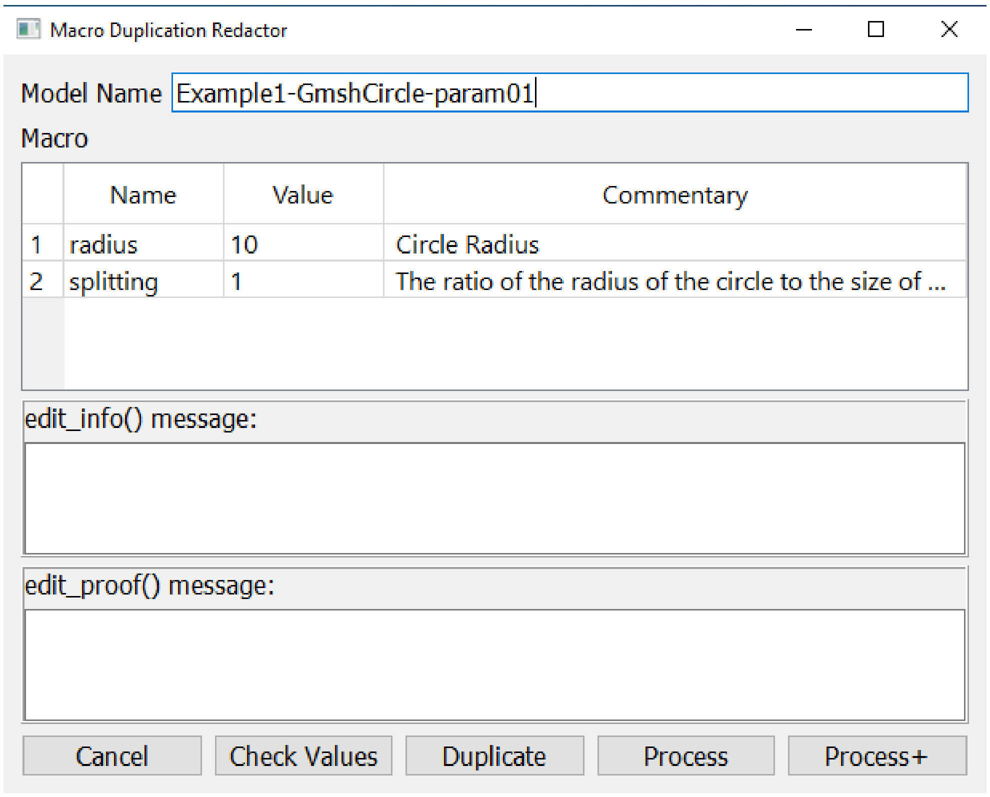
3.9 File extensions.

3.9.1 The program uses file extensions that allow the program to correctly select handlers and run the appropriate processes, ensuring correct operation with various types of data.

3.9.2 3.9.2 Project files can have any extension. For expansion **.spad** is configured by the association to launch the SimPad program and open the specified project file in the editing interface.

3.10 Duplication of the processed file.

3.10.1 The duplication window looks like this



Model Name Example1-GmshCircle-param01

Macro

	Name	Value	Commentary
1	radius	10	Circle Radius
2	splitting	1	The ratio of the radius of the circle to the size of ...

edit_info() message:

edit_proof() message:

3.10.2

3.10.3 The **Model Name** field will contain the editable model name.

3.10.4 The **Macro** field displays an editable table of macro variables consisting of three columns: **Name** – the variable name that cannot be changed, **Value** – the variable value that can be changed, **Commentary** – the comment to the macro variable that cannot be changed.

3.10.5 In the **edit_info() message field**, a message is output from the **edit_proof** method, in which the values of the macro variables are checked.

3.10.6 In the **edit_info() message field**, a message from the **edit_info** method is output, with information about the variables.

3.10.7 **Cancel** button – closes the duplication window.

3.10.8 **The Check Values** button checks the values of macro variables. When clicked, it transfers the current values from the macro variables table to the edit_proof and edit_info methods, followed by the output of the messages from them

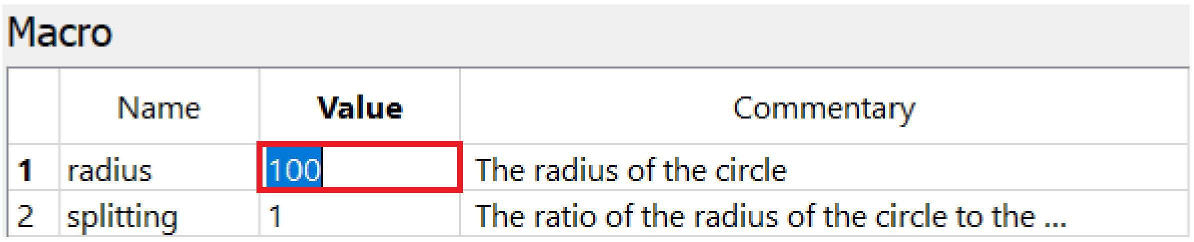
3.10.9 **The Duplicate** button starts the macro substitution. When clicked, it first starts checking the values of the macro variables, and then macro substitution of the values of the variables specified in the table into the processed file, followed by saving it and opening it in the text editing window.

3.10.10 **The Process** button duplicates and starts the default handler specified in the file. If the default handler is not set, the handler selection window is displayed.

3.10.11 **The Process+** button duplicates and launches a dialog box with the choice of a handler.

3.10.12 To edit the value of a variable, double-click with the left mouse button on the field with the value and make the necessary changes.

3.10.13


Macro

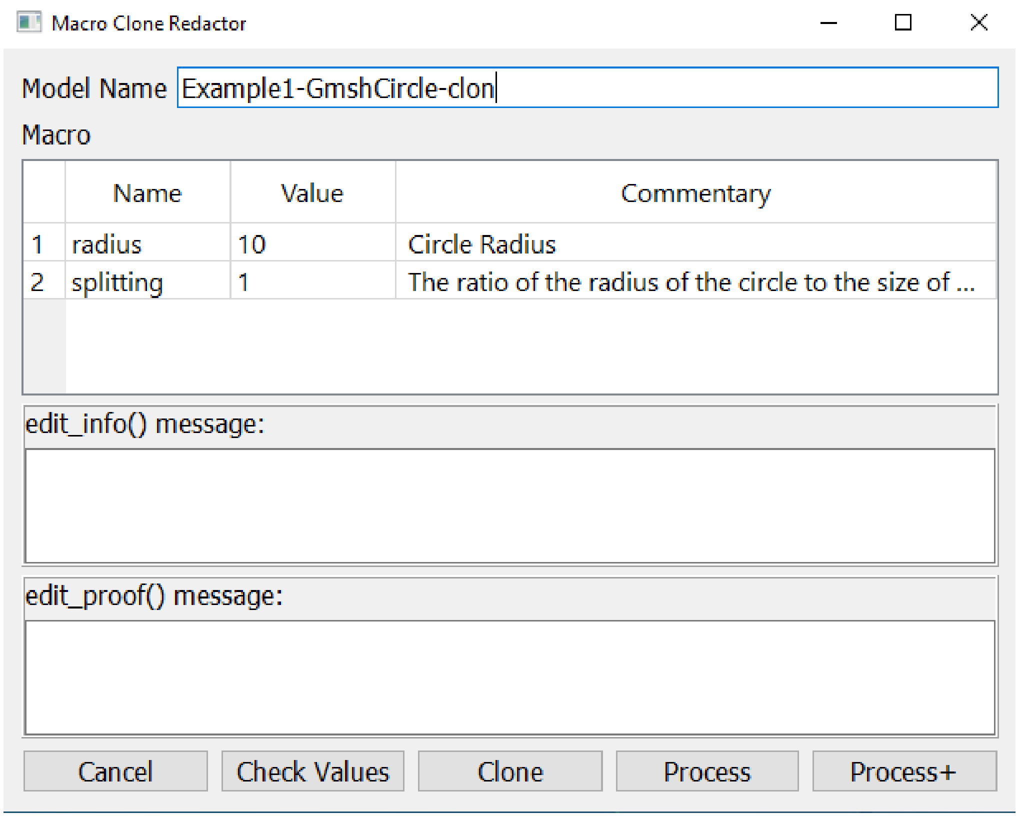
	Name	Value	Commentary
1	radius	100	The radius of the circle
2	splitting	1	The ratio of the radius of the circle to the ...

3.10.14 To duplicate, click the **Duplicate** button. If there are no @@datafile data file blocks in the text of the processed file, or the mono type is set as the processing type, the processing result will be a single file. Strings that are not included in the simPad block sets will be processed by macro substitution and added to the resulting file. If the text of the processed file contains blocks of @@datafile data files and the type of processing is set to **multi or auto**, then @@datafile blocks will be processed during duplication, and the result of processing will be a set of several files corresponding to the number of specified @@datafile blocks. In this case, the lines outside the blocks are not processed.

3.10.15 You can perform duplication by clicking the **Process and Process+** buttons. In addition to duplication, the file will be processed by the default handler or with the option to select, depending on the button pressed.

3.11 Cloning of the processed file.

3.11.1 The cloning window looks like this.


Macro Clone Redactor

Model Name: Example1-GmshCircle-clon

Macro

	Name	Value	Commentary
1	radius	10	Circle Radius
2	splitting	1	The ratio of the radius of the circle to the size of ...

edit_info() message:

edit_proof() message:

3.11.2

3.11.3 The cloning window is identical to the duplication window and has the same functionality. The only difference is the **Clone** button, instead of the **Duplicate** button.

3.11.4 The fundamental difference between cloning and duplication is that when cloning, the created file is created and updated, and when duplicating, a new one is created each time. Otherwise, the cloning process completely repeats the duplication process.

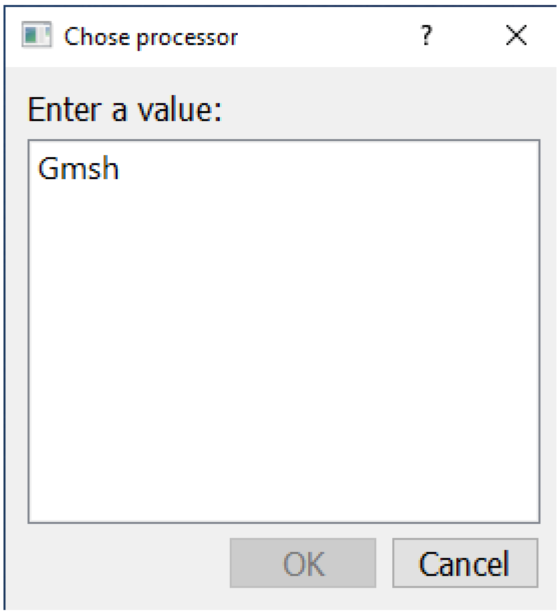
3.11.5 To perform cloning, click on the Clone button. Cloning with the presence of @@datafile blocks is similar to duplication.

3.12 Calculation of the processed file.

3.12.1 The program implements the possibility of calculating the processed file in two ways, single calculation (calculation session) **Single Calculation** and batch calculation (batch session) **Packet Calculation**.

3.12.2 To perform a single calculation, select the **Single Calculation** item in the Calculation menu item or use the corresponding button on the toolbar. If the default handler is not specified in the text of the processed file in the **@proc** block, when starting the calculation, a dialog box for selecting a processor will appear on the screen.

3.12.3 The window for selecting the **Choose processor** looks like this



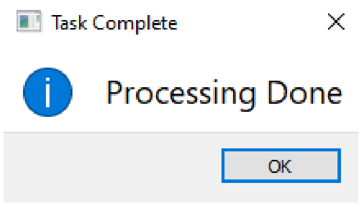
3.12.4

3.12.5 In **the Enter a value** field, the handlers available for use are displayed.

3.12.6 **The Cancel** button closes the handler selection dialog box.

3.12.7 To start the calculation, in the **Enter a value** field, select the necessary handler by clicking on it once with the left mouse button, and then click OK.

3.12.8 After successful calculation, a window with a message about successful completion will be displayed on the screen


Task Complete

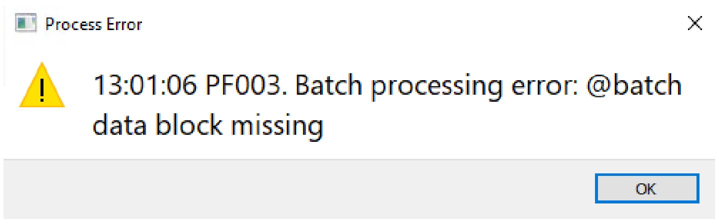
Processing Done

3.12.9

3.12.10 To start the batch **Calculation**, select the **Packet Calculation** item in **the Calculation** menu item or use the appropriate button on the toolbar. If the default handler is not specified in the text of the processed file in the **@proc** block, when starting the calculation, a dialog box for selecting a processor will appear on the screen.

3.12.11 The window for selecting a handler for batch calculation is similar to the window from a single calculation and has identical functionality.

3.12.12 If the block of data sets for batch processing **@batch** is not specified in the text of the processed file, then when trying to start batch calculation, an error window will appear on the screen, batch processing will not be possible.


Process Error

13:01:06 PF003. Batch processing error: @batch data block missing

3.12.13

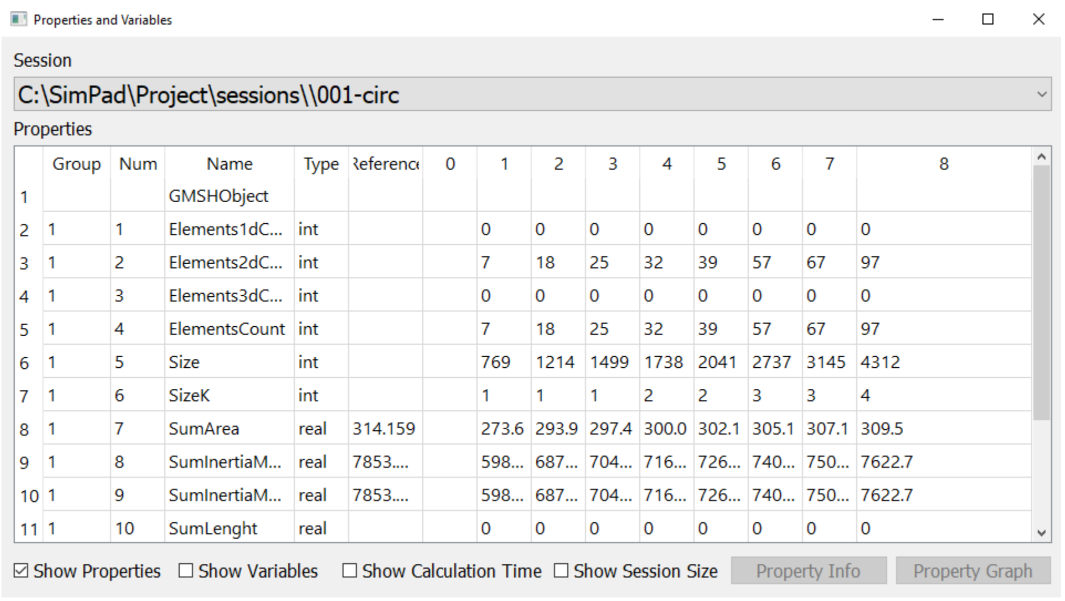
3.13 Viewing the calculation results

3.13.1 The program provides the ability to view the calculation results of processed texts in the form of tables, graphs, images and using a viewer.

3.13.2 **Viewing the results in the form of a table.**

3.13.3 To open the window with the results of the **Properties and Variables** calculation, select the Properties item in the Results menu or use the button on the toolbar.

3.13.4 The window for viewing the results of the Properties and Variables calculation in the form of a table looks like this

3.13.5 

Properties and Variables

Session

C:\SimPad\Project\sessions\\001-circ

Properties

	Group	Num	Name	Type	Referenc	0	1	2	3	4	5	6	7	8
1			GMSHObject											
2	1	1	Elements1dC...	int			0	0	0	0	0	0	0	0
3	1	2	Elements2dC...	int			7	18	25	32	39	57	67	97
4	1	3	Elements3dC...	int			0	0	0	0	0	0	0	0
5	1	4	ElementsCount	int			7	18	25	32	39	57	67	97
6	1	5	Size	int			769	1214	1499	1738	2041	2737	3145	4312
7	1	6	SizeK	int			1	1	1	2	2	3	3	4
8	1	7	SumArea	real	314.159		273.6	293.9	297.4	300.0	302.1	305.1	307.1	309.5
9	1	8	SumInertiaM...	real	7853....		598...	687...	704...	716...	726...	740...	750...	7622.7
10	1	9	SumInertiaM...	real	7853....		598...	687...	704...	716...	726...	740...	750...	7622.7
11	1	10	SumLenght	real			0	0	0	0	0	0	0	0

☑ Show Properties ☐ Show Variables ☐ Show Calculation Time ☐ Show Session Size

3.13.6 **The Session** drop-down list is set for the last processing session, if desired, you can select previous sessions.

3.13.7 **The Properties** table consists of six main columns. The Group column indicates the group number of the object to which the property belongs. In the Num column, the sequence number for properties and variables is automatically set. The Name column specifies the name of the properties and variables. The Type column specifies the data type for properties and variables. The Reference column displays the references specified for the calculated values. The column with the number zero displays the results of the post-processors. Then there are numbered columns starting from one, which display the calculated values of properties or values of variables.

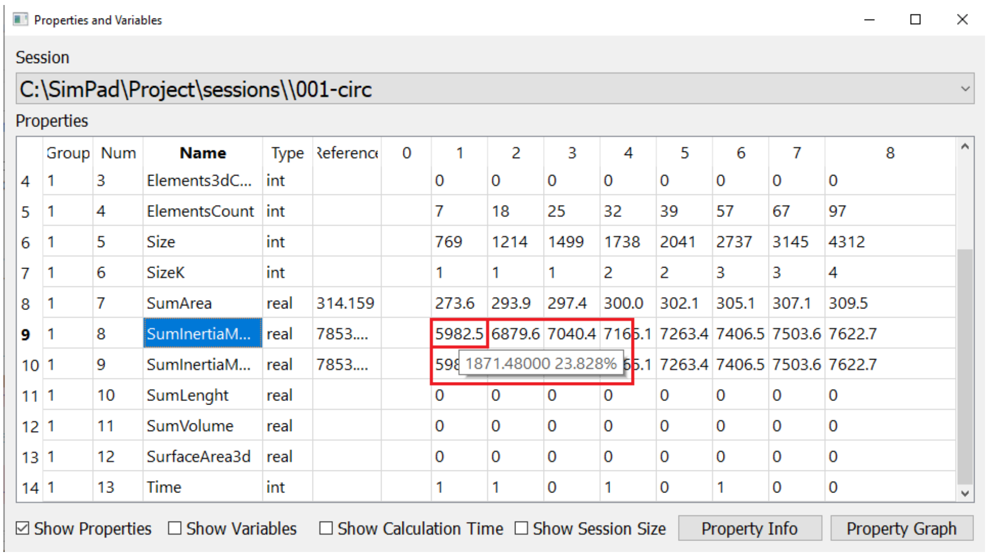
3.13.8 **The Show Properties** checkbox enables or disables the display of properties in the Properties table.

3.13.9 **The Show Variables** checkbox enables or disables the display of variables in the Properties table.

3.13.10 **The Show Calculation Time** checkbox enables or disables the display of the calculation time in the Properties tab

3.13.11 **The Show Session Size** checkbox enables or disables the display of the session folder sizes in the Properties tab.

3.13.12 To view the absolute and relative deviations from the reference set for a certain calculated value, hover the mouse cursor over a cell of the calculated value, after which percentage deviations will be displayed in the hint of this cell. If the relative deviation is greater than the maximum value specified in the configuration file (by default, 100%), it is not displayed.


Properties and Variables

Session

C:\SimPad\Project\sessions\\001-circ

Properties

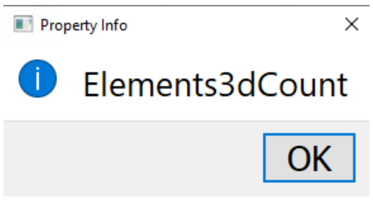
	Group	Num	Name	Type	Reference	0	1	2	3	4	5	6	7	8
4	1	3	Elements3dC...	int			0	0	0	0	0	0	0	0
5	1	4	ElementsCount	int			7	18	25	32	39	57	67	97
6	1	5	Size	int			769	1214	1499	1738	2041	2737	3145	4312
7	1	6	SizeK	int			1	1	1	2	2	3	3	4
8	1	7	SumArea	real	314.159		273.6	293.9	297.4	300.0	302.1	305.1	307.1	309.5
9	1	8	SumInertiaM...	real	7853....		5982.5	6879.6	7040.4	7165.1	7263.4	7406.5	7503.6	7622.7
10	1	9	SumInertiaM...	real	7853....		598	1871.48000 23.828%		65.1	7263.4	7406.5	7503.6	7622.7
11	1	10	SumLenght	real			0	0	0	0	0	0	0	0
12	1	11	SumVolume	real			0	0	0	0	0	0	0	0
13	1	12	SurfaceArea3d	real			0	0	0	0	0	0	0	0
14	1	13	Time	int			1	1	0	1	0	1	0	0

☑ Show Properties ☐ Show Variables ☐ Show Calculation Time ☐ Show Session Size

3.13.13

3.13.14 **The Property Info** button displays information about the property. To display information, left-click on the cell with the name of the property, and then click on **the Property Info** button. A window with information about the property will appear on the screen.

3.13.15 The window with information about the property looks like this

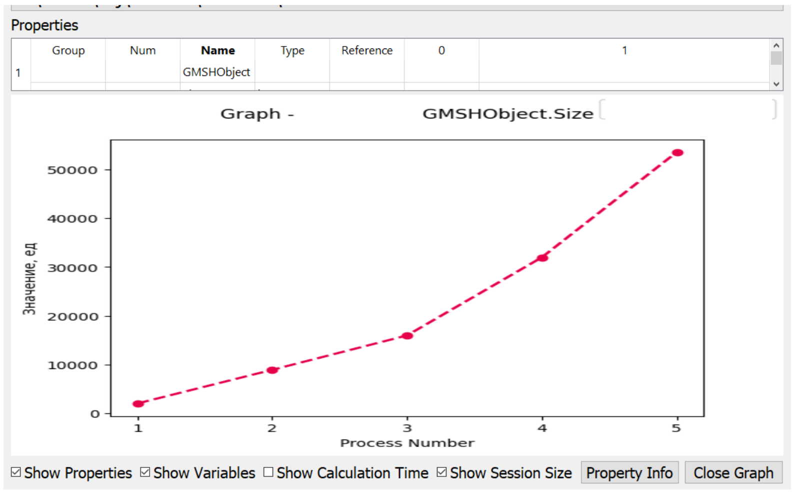

Property Info

Elements3dCount

3.13.16

3.13.17 **The Property Graph** button displays a graph of the values of the selected property in **the Properties and Variables** window. To display the graph, left-click on the cell with the name of the property, and then click on **the Property Graph** button. **The Properties and Variable** window displays a graph based on the calculated property values.

3.13.18 **The Properties and Variables** window with the graph looks like this


Properties

	Group	Num	Name	Type	Reference	0	1
1			GMSHObject				

☑ Show Properties ☑ Show Variables ☐ Show Calculation Time ☑ Show Session Size

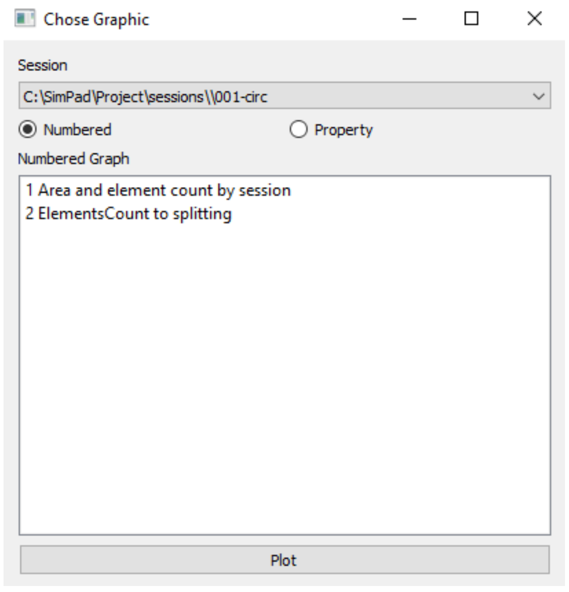
3.13.19

3.13.20 Clicking on the Close Graph button, the graph will close

3.13.21 **Viewing the calculation results in the form of graphs**

3.13.22 To open the window with the **Graphchose** graph selection, select the **Graphic** item in the **Results** menu or use the corresponding button on the toolbar.

3.13.23 The Graph select window **Graphchose** looks like this

3.13.24 
Chose Graphic

Session: C:\SimPad\Project\sessions\\001-circ

[✓ Numbered] Property

Numbered Graph

1 Area and element count by session
2 ElementsCount to splitting

3.13.25 **The Session** drop-down list allows you to select the calculation session for which you want to build graphs

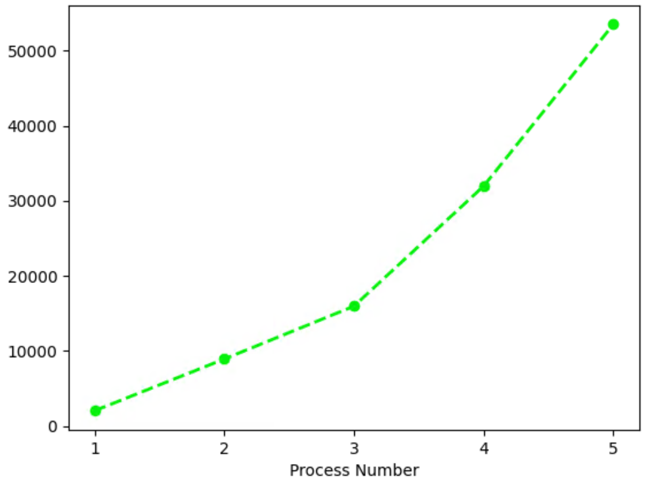
3.13.26 **Numbered** button, if the checkbox is selected on the **Numbered** button, all numbered graphs specified in the source file will be displayed in the display field of graphs available for construction.

3.13.27 **Property** button, if the checkbox is selected on the **Property** button, graphs of property values and their references will be displayed in the graph display field.

3.13.28 **The Plot** button builds the selected graph when clicked.

3.13.29 To build a graph, select the type of graph, numbered graphs or graphs of values, then select the necessary graph in the graph selection field and click **the Plot** button.

3.13.30 A window with a schedule will open



3.13.31

3.13.32 **Viewing images**

3.13.33 To open **the Image Viewer** Image Viewer window, select the Image item in the **Results** menu or use the corresponding button on the toolbar.

3.13.34 **The Image Viewer** window looks like this

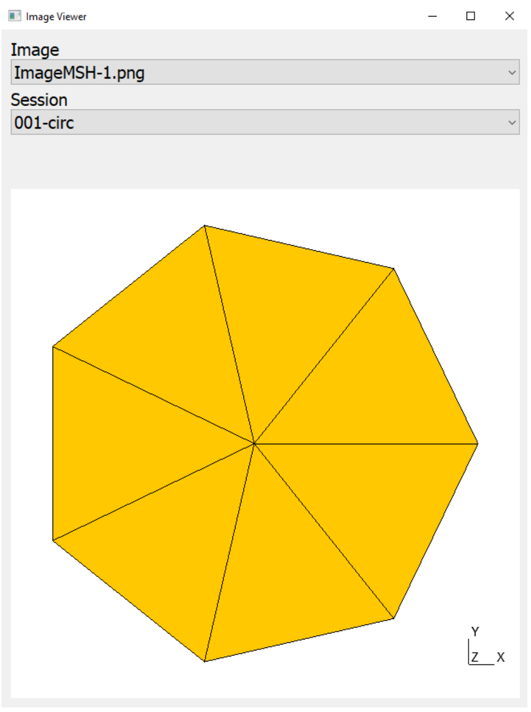


Image Viewer

Image
ImageMSH-1.png

Session
001-circ

3.13.35

3.13.36 In the **Image** drop-down list, you can select an image.

3.13.37 In the **Session** drop-down list, you can select a billing session.

3.13.38 If the selected session is a batch session, a new **Result** drop-down list will appear in the image display window, in which you can select an image for a specific set of data.

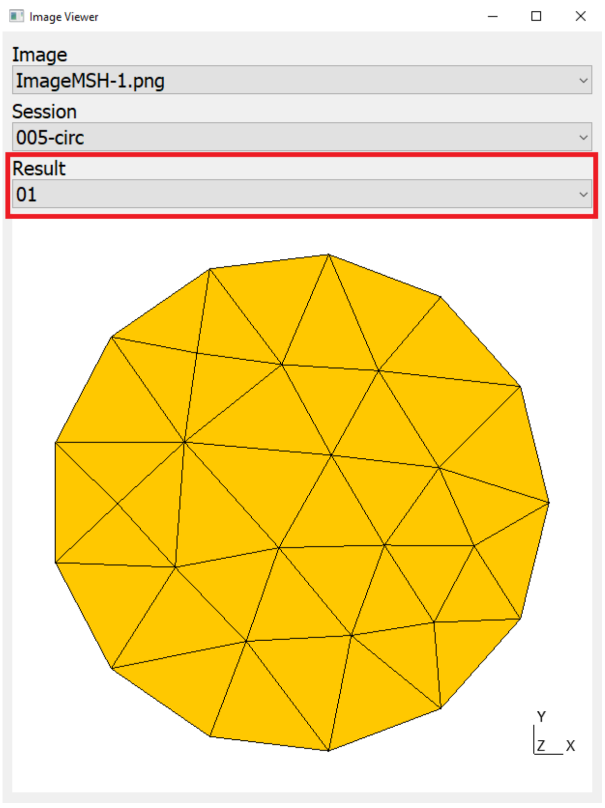


Image Viewer

Image
ImageMSH-1.png

Session
005-circ

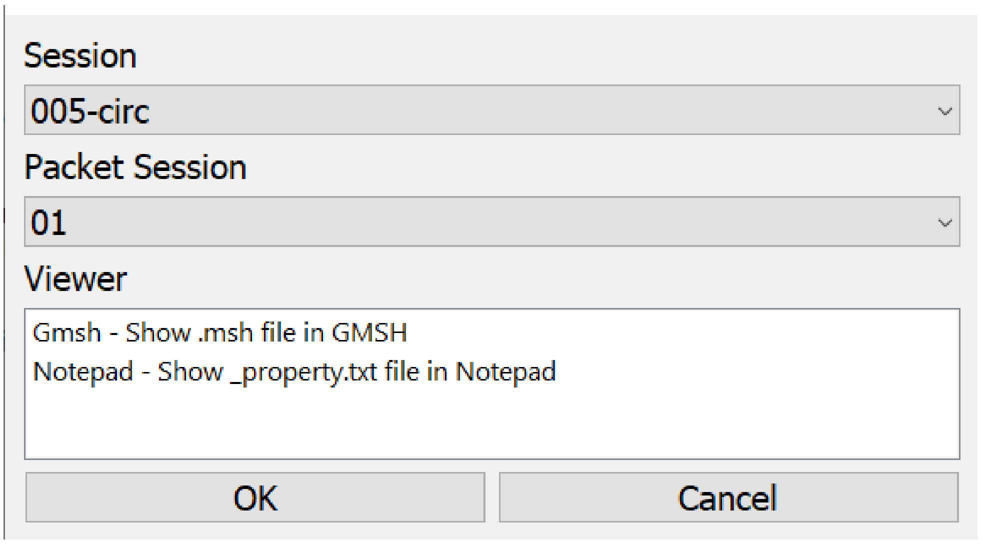
Result
01

3.13.39

3.13.40 **Viewers.**

3.13.41 To open the viewer selection window, select the **Viewers** item in the **Results** menu or use the corresponding button on the toolbar.

3.13.42 The viewer selection window looks like this


Session
005-circ
Packet Session
01
Viewer
Gmsh - Show .msh file in GMSH
Notepad - Show _property.txt file in Notepad

3.13.43

3.13.44 In the **Session** drop-down list, you can select a billing session.

3.13.45 In the **Packet Session** drop-down list, you can select a specific data set to view in the viewer.

3.13.46 The **Viewer** field displays the viewer available for use.

3.13.47 To open the viewer, left-click on the required viewer in the **Viewer** field, and then click **OK.** The viewer will be launched.

4. Macro markup.

4.1.1 The macro processor data blocks are used by the macro processor to configure and perform data processing tasks. Each data block performs a specific function and contains instructions that define exactly how the input data should be processed and what output results should be obtained.

4.1.2 Blocks begin with a special character **@** or **@@**, after which the block name is indicated without a space.

4.1.3 Identifiers starting with a single **@** character specify a block nested in a block starting with an identifier with two characters **@@**

4.1.4 The presence of an unknown identifier at the beginning of the line, starting with a special character, should cause a warning to be displayed during processing.

4.1.5 Other characters can be specified in the program call keys.

4.1.6 The block name can be either uppercase or lowercase – for example, **@@simpad** and/or **@@SIMPAD**. The syntactic processing of names in different registers is the same, but setting syntax highlighting may be different and make some sense to the user.

4.1.7 **The @@simpad block**

4.1.8 **@@simpad** defines the beginning of a set of data blocks for macro substitution. Two parameters type and ext are set in the block.

4.1.9 **The type=Processing type** parameter specifies the type of data processing that affects how the data blocks in the file will be processed.

4.1.9.1 **auto** – the type of processing is determined by the presence of @file blocks in the text (by default)

4.1.9.2 **mon**o is a single–file type of processing, @@datafile blocks are ignored in the file, the result of processing is a single file. All lines that are not included in the simPad block sets are processed by macro substitution and added to the resulting file

4.1.9.3 **multi**–file type of processing - @@datafile blocks are processed in the file, the result of processing can be a set of several files. lines of text outside the @@datafile blocks are not processed.

4.1.10 **ext=Main extension** – used to call handlers and viewers. If not specified, the extension of the first block of the data file is used.

4.1.11 **Nested block @job.**

4.1.12 **The @job** block is used to specify a task description and may contain multiline text. It specifies three parameters id, name and tag

4.1.13 **id=designation** – a short designation of the task, a literal without spaces. As a rule, Latin letters and numbers are used. Only characters that are allowed in file names can be used.

4.1.14 **name=the name of the task**. If it contains spaces, then in quotes

4.1.15 **tag1=tag1** is an arbitrary code. It is used to select handlers, viewers, etc. in cases. Each tag must be numbered.

4.1.16 The task designation is used in the names of the session folders.

4.1.17 The specified **@job** block should look like this:

@job id=Designation **name=**Task name **tag1=**Tag **is a multi-line text describing the task being solved**

4.1.18 **Nested block @doc.**

4.1.19 The **@doc** block is used to specify a multiline text with a description of the macro substitution. It sets three parameters num, tag and dofile.

4.1.20 **num= File number**, if not specified, it is equal to the previous one plus 1.

4.1.21 **tag=tag** – integer or literal without spaces

4.1.22 **dofile=code** – the code for creating a file in the session folder. 1-yes, 0-no. the name of the file being created doc-N-T.txt N is a three-digit file number starting from 001, T is the tag.

4.1.23 The specified **@doc** block should look like this:

num=File number **tag**=tag **dofile**=code - **multiline text of the description of the macro substitution**

4.1.24 **Nested @proc block**

4.1.25 The **@proc** block specifies a list of handlers that can be used to process data. The first handler in the list is the default handler. If there is no such processor in the program instance, then when trying to calculate the processed file, a dialog for selecting handlers is displayed - handlers specified in **@proc** and handlers corresponding to the file extension.

4.1.26 **default – the default handler number**. If it is not specified or there is no such handler in the program instance, then when the handler is started, the default handler selection dialog is displayed - the handlers specified in @proc and handlers corresponding to the file extension, for configured in the program instance

4.1.27 the specified **@proc** block should look like this:

@proc Handler name number handler description

4.1.28 **Handler name** is a literal without spaces corresponding to the name of the handler configured in the program instance

4.1.29 **Nested block @format.**

4.1.30 **The @format** block is used to set data conversion parameters such as delim, comm, info, liner, inline and proc

4.1.31 **delim="Delimiter string".** The parameter specifies the delimiter characters used to search for macro variables when inserting macro variables. When specifying a space, the tab is also used as a separator. By default, the delimiters are space, comma, semicolon. For example, to specify a space, dollar, comma, and semicolon, the command will look like **@delim " $,;"** . Several identical separator characters can be specified in a line. For example, "$, ;"

4.1.32 **comm="Comment symbols",** the parameter specifies comment literals in the lines of the processed file - a list of literals separated by a space – the literal of the inline comment (the part of the line from the comment symbol to the end of the line is not processed), the literal of the beginning of the comment block, the literal of the end of the comment block. For example, **@comm "## /# #/".** By default – **"// /* */"/ .** Comments can be placed anywhere in the source file, including in the **sections @macro, @math and @func, etc**. Python comments are also used in the **@math and @func** sections.

4.1.33 **info=1 or 2.** When setting this command to the beginning (info=1) or to the end (info=2) the file with the results of the macro substitution displays blocks of lines:

4.1.33.1 processing descriptions: with the date and time of the macro substitution, the name of the program version, the name of the source file, time and size in bytes

4.1.33.2 editable macro variables – number in order, in the order of the task in the file, name and value of the macro variable, description

4.1.33.3 calculated macro variables – number in order, in the order of the task in the file, name and value macro variable, description (comment at the end of the line)

4.1.33.4 function output edit_info() – function output lines

4.1.34 **liner="Liner sign"**– output of information about macro substitution - before the lines in which the macro substitution was performed, they are displayed in comments with the liner sign and with a string containing macro-variable and inline expressions. Example of liner=" ->"

4.1.35 **inline="limiters"** - limiters of inline arithmetic expressions – a pair of literals of 1 or 2 characters separated by spaces. Examples - "{ }", "{< >}", "</ />". The default is "{ }". An inline expression is an arithmetic formula in Python using constants, meta-variables, and mathematical functions defined in func, func2, and func3.

4.1.36 **proc=0 or 1 or 2** – output comment lines with information about @proc handlers to the beginning of the file. These comment lines can be processed by SimPad to run default handlers for the file opened in the interface.

4.1.36.1 0 – information is not output

4.1.36.2 1 – information is output in the first file of single-file and multi-file mode

4.1.36.3 2 – information is displayed in all files

4.1.37 **@@macro block.**

4.1.38 3.14.38 The @@macro block is used to set macro variables and macro substitutions using scripts.

4.1.39 3.14.39 There may be several sections in the file @edit @enum. The order of setting variables in sections corresponds to the order in which they are displayed in the program interface.

4.1.40 **Nested block @edit**

4.1.41 **The @edit** block is a section of editable numeric and string macro variables. There may be spaces between words in the lines, but there may not be quotation marks. After the value of the variable, the variable identifier can be specified in quotation marks, and after the identifier, a description. The identifier may be missing, i.e. the identifier name may be empty

4.1.42 The specified **@edit** block should look like this:

"Variable name" "Variable value" "identifier" variable description

4.1.43 **Nested block @enum**

4.1.44 The **@enum** block is a section of enumerated numeric and string macro variables. In the interface, the variable is edited using a drop-down list, which displays a description of the value, and if it is not set empty, the value is displayed. After the variable name, a sequence of variable values is set, as well as a description of the value in quotation marks. All values must be of the same type – integers or real numbers or strings. After the value of the variable, the variable identifier can be specified in quotation marks, and after the identifier, a description. The identifier may be missing, i.e. the ID name can be empty. By default, the first value from the list is used.

4.1.45 The specified **@enum** block should look like this:

"Variable name" value1="Description of value 1" value2="Description of value 2" ... "identifier" variable description

4.1.46 **Nested block @math**

4.1.47 Block **@math** is a section of calculated macro variables of numeric and string type.

4.1.48 After specifying an arithmetic or string expression, the name of the record in quotation marks can be specified, after which a comment can be specified

4.1.49 The specified **@math** block should look like this:

"Variable name" (arithmetic or string expression) "record name (in quotation marks)" (comment)

4.1.50 **Nested block @mtype.**

4.1.51 The **@mtype** block is used to define the types of macro variables that are used in the **@edit, @enum, and @math** sections.

4.1.52 Types of macro variables:

4.1.52.1 **int** is an integer

4.1.52.2 **float** is a real number

4.1.52.3 **str** is a string in quotation marks, including a string with spaces, but without internal quotation marks. Spaces at the beginning and end of lines are discarded. Examples: "AB CD" "123"

4.1.52.4 **astr** is a quoted string containing an array of numbers and/or quoted strings. The array is enclosed in square brackets, the objects are separated by commas. Example: "[1, 2]" "[[1,1], [1,2]]"

4.1.52.5 **jstr** is a quoted string containing a **JSON** object in curly brackets. Example "{ "A": 1 }"

4.1.53 The error is used to round (- in the edit window, in the output functions) the calculated value of the variable, taking into account the physical meaning of the value.

4.1.54 Error = Absolute error + (Relative error * of the variable value)

4.1.55 **The number of significant digits** is the rounding parameter for representing the number in the interface (– in the editing window, in the output functions), if the error value is set to 0.

4.1.56 **Additional digits** – additional digits to the significant ones that are output to the value of the variable during macro substitution in data files. The zeros on the right are discarded.

4.1.57 Calculations with variables are performed without rounding.

4.1.58 Macro variables not described in this section are of the **real** type and are rounded to the values specified in the **itol, rtol** parameters.

4.1.59 **itol=abs,rel,dig,adig** – absolute and relative error of integers. By default, **itol=1.0.0.0.2**

4.1.60 **rtol=abs,rel,dig,adig** - absolute and relative error , the number of significant digits of real numbers. By default, **rtol=0.0,0.0,4,2**

4.1.61 **Nested block @param.**

4.1.62 The **@param** block is used to set macro variables, which are processing parameters – the keys for launching the bat file of the handler. each line contains the name of the macro variable, then a comment can be set. Transferring keys to the bat file after the -param key in the order of setting the names of macro variables in this section

4.1.63 **Nested block @mbat.**

4.1.64 The **@mbat** block is used to specify macro variables that are used for macro substitution in dynamic handler bat-files. If there is no section, then all macro variables are applied.

4.1.65 **Nested block @func.**

4.1.66 The **@func** block is a section of macro–variable functions. To get the values of macro variables, **getMacro()** function calls are used.

4.1.67 Functions can use calls to SimPad API functions for which the macroAPI module is provided.

4.1.68 **Nested block @func2.**

4.1.69 Block **@func2** is a section of macro–variable functions. After the function declaration lines in the section, the variable declaration lines in the **@macro** sections are automatically added and values assigned to them by calls to **getMacro().**

4.1.70 **Nested block @func3.**

4.1.71 Block **@func3** is a section of macro–variable functions. After the function declaration lines, the variable declaration lines in the **@macro and @math** sections are automatically added to the section and values are assigned to them by calls to **getMacro().**

4.1.72 Functions can use calls to the SimPad API functions, for which the macroAPI module is provided.

4.1.73 **The @@post block.**

4.1.74 The **@@post** block is used to start a set of data blocks that are transmitted to the postprocessors.

4.1.75 Postprocessors create or add a file **_property.txt** with **JSON** and/or image data in the _IMG subfolder of the session folder.

4.1.76 **Nested block @ref.**

4.1.77 The **@ref** block is used to describe references.

4.1.78 A _ref,txt file with a description of the references is saved to the sessions folder.

4.1.79 The reference error is used to round off properties when displaying calculated property values in the table, unless otherwise specified in the property accuracy parameters in the postprocessor processing file.

4.1.80 The specified @ref block should look like this:

"Property name" "Property control value" "Reference error"

4.1.81 **Block @@batch.**

4.1.82 The **@@batch** block defines the beginning of a block of data sets for batch processing.

4.1.83 The same macro variables cannot be set in different sets of values.

4.1.84 Combinations of macro-variable values are created from sets of macro-variable values, that is, specifying several sets of variables sets a multidimensional table of possible combinations of values and a corresponding tree-like system of nested sessions.

4.1.85 Example – in set 1, variable A takes the values A1 and A2, in set 2, variable B takes the values B1, B2, B3 and C values C1 C2 C3

4.1.86 Then the following combinations of variable values for calculations are obtained

	Set 2(1)	Set 2 (2)	Set 2 (3)
Set 1 (1)	A1 B1 C1	A1 B2 C2	A1 B2 C3
Set 1 (2)	A2 B1 C1	A2 B2 C2	A2 B3 C3

4.1.87 Session structure.

4.1.88 Batch session.

4.1.88.1 001 Batch session for Typeset1 (1)

4.1.88.1.1 001 Calculation session for Typeset2 (1)

4.1.88.1.2 002 Calculation session for Typeset2 (2)

4.1.88.1.3 003 Calculation session for Typeset2 (3)

4.1.88.2 002 Batch session for Typeset1 (2)

4.1.88.2.1 001 Calculation session for Typeset2 (1)

4.1.88.2.2 002 Calculation session for Typeset2 (2)

4.1.88.2.3 003 Calculation session for Set2 (3)

4.1.89 **Nested block @list.**

4.1.90 The **@list** block is a section with a list of values of the specified macro variables.

4.1.91 The specified @list block should look like this:

"Variable name" "List of values"

4.1.92 Nested block **@loop**

4.1.93 Block **@loop** is a section with parameters of cyclic values of the specified macro variables. There may be several consecutive cycles defined by the step size and the number of steps.

4.1.94 The number of tasks to be created in the package is the maximum number of values of any variable in the **@list and @loop** sections. If the number of values set for a variable is less than the maximum, the last set value is used for the unset values.

4.1.95 The specified **@loop** block should look like this:

"Variable name" "Initial value" step1 Number of steps 1 …

4.1.96 **The @@bpost block.**

4.1.97 The **@@bpost** block is a data section for the postprocessor of batch processing results.

4.1.98 **Nested block @chart.**

4.1.99 The **@char** block is used to describe dependencies that should be displayed on graphs. To visualize graphs, a **Python** script is created using the **matplotlib** library, which returns a graph image in **SVG** or **IMG** format. This image is then displayed in the user interface. The block contains 2 parameters **(num and name)**

4.1.100 **num=chart number**, the parameter specifying the chart number.

4.1.101 **name=description of the Graph**, a parameter specifying a description for the graph.

4.1.102 The specified **@char** block should look like this:

type minimum Value maximum Value list of Properties "group name" description of the Group

4.1.103 **type** – type of the group of values. For each group of abscissas, a separate axis with marks is shown on the graph.

4.1.103.1 **10 -** ordinate. only one property

4.1.103.2 **20** can be set – the abscissa. the minimum and maximum values of the range are not set

4.1.103.3 **21** - abscissa. the minimum and maximum values of the displayed range

4.1.103.4 **22** - abscissa. the minimum values of the displayed range are set

4.1.103.5 **23**- abscissa. the maximum values of the displayed range

4.1.103.6 **30..33** – the abscissa corresponds to types 20..23. In addition to the abscissa value, a graph of approximation by the power function A + B* ordinate C is output

4.1.104 **Example:**

@chart 1 "Example chart"

10 0 0 P "calculation parameter, meters"

22 0.0 0 T "calculation time, minutes"

20 0 0 X Y Z "displacement, mm"

4.1.105 **The @@datafile block.**

4.1.106 The **@@datafile** block defines the extension and content of the output file. Four parameters are set in the block **(num, key, ext and comm).**

4.1.107 For the @@datafile instruction, you can also set synonyms – instructions of the form @@datafile_N, where N are numbers from 1 to 9.

4.1.108 **num=File number**, the parameter specifies the file number.

4.1.109 **key=key**, sets the key, if this parameter is set, then when passing the file name to the processor, this additional key is indicated before the file name

4.1.110 **ext="File extension"** The parameter specifies the extension for the source, resulting file.

4.1.111 **comm="Comment symbol",** the parameter specifies a comment to a block of the data file.

4.1.112 For Each section of the @@datafile, when cloning or duplicating, a data file is created with the form **"Workable file name-NN-clone.File Extension" or "The name of the employee is your file-NN-paramMM.File extension".**

4.1.113 **NN** is the file number, corresponding to the ordinal number of the **@@datafile** data section in the source file, starting from 01.

4.1.114 The comment symbol overrides the comment symbols specified in **@format** for the corresponding file being created.

4.1.115 **Filenumber** – the serial number of the file in the list of files transmitted to the handler in the startup keys. If not specified, it is equal to the number of the previous file, increased by 1. If the number of the first file is not specified, then it is equal to 1.

4.1.116 Block **@@end** – the end of a set of data blocks. rows between the end of a set of data blocks to the beginning of the next set are not processed in the multi processing mode.

4.1.117 **The @label block.**

4.1.118 The @label block is a string label for use in functions for adding strings to API functions ap-pend_line (numbers, ...). The **num** parameter is set in the block.

4.1.119 **Function parameters** - a list of strings and numbers from which the inserted string is formed. To form an inserted string, you can specify the separator to use by calling the **API line_delimeter** function (separator). The default separator is a space " ". When the **append_line** function (label numbers, ...) is called several times, the corresponding group of lines is inserted instead of the label.

4.1.120 The specified **@label** block should look like this: **@label num=Label numbers**

4.1.121 **Block @if.**

4.1.122 The **@if** block is a conditional execution command. Strings, before the **@endif** operator. The lines up to the following **@endif** operator are processed if the condition specified by the parameters is true. Three parameters are set in the block (macro, value and operator)

4.1.123 **@if** commands can be nested.

4.1.123.1 **macro** is the name of the macro variable.

4.1.123.2 **value** – the value being compared.

4.1.123.3 **operator** – comparison operator.

4.1.123.4 **"="** is equal to.

4.1.123.5 **">"** - more.

4.1.123.6 **"<"** - less.

4.1.123.7 **"=!"** is not equal.

4.1.123.8 **">="** - greater than or equal to.

4.1.123.9 **"<="** - less than or equal to.

4.1.124 The specified **@if** block should look like this:

@if macro="Variable name" value="Comparability value" operator="="

4.1.125 **Block @endif.**

4.1.126 Block @endif – completion of the conditional execution block.

4.1.127 **The @include block.**

4.1.128 The **@include** block inserts blocks of text that do not contain SimPad commands. The absence of commands is checked during insertion. Two parameters are set in the block **(file or lib).**

4.1.129 **file=File path** – the path to an arbitrary file.

4.1.130 **lib=Filename** – the name of the file in the library folder of the standard program files.

5. Syntax highlighting styles.

5.1 Highlighting styles

5.1.1 Syntax highlighting styles are used to highlight various elements of code and text in an application. They are configured using text files containing regular expressions and font settings. These files are divided into sections, each of which has its own identifier starting with the "@" character.

5.1.2 **Syntax highlighting style (syntax file)** is a text file with lines for specifying regular expressions of syntax blocks and font parameters.

5.1.3 Lines in the record block, a list of fields separated by spaces

5.1.4 The identifiers of the records are unique, the uniqueness must be checked when parsing

5.1.5 Blocks are highlighted in order of priority. If the priority is the same, the first block found is in the order of setting records and blocks in the list of records in the DINAMIC and STATIC sections. The DINAMIC section is considered before the STATIC section.

5.1.6 If there are font parameters not specified in the block style, then the value of this field is assumed for the first block of the next priority. The maximum number of iterations of searching for unspecified style elements is specified in the configuration file, by default 3.

5.1.7 If an undefined style value is not found, the default value specified in the configuration file is used.

5.2 Blocks and fields of records.

5.2.1 The **@LITERAL** block is a section for specifying literals. The section includes:

5.2.1.1 The literal identifier

5.2.1.2 The bracket group. An integer indicating the group of regular expression matches enclosed in parentheses that will be highlighted. (Default is 0 – all matches)

5.2.1.3 List of regular expressions.

5.2.2 Example: You need to highlight the type string if there is an expression @@simpad in the line in front of it.

5.2.2.1 Regular expression: **(?<=@@simpad)\s(.*)(type)(?=\s*\=) (?<=@@simpad) – positive lookbehind** – checking the condition before the highlighted groups.

5.2.2.2 **\s** – required space

5.2.2.3 **(.*)** – bracket group 1 – any characters after the condition and before the bracket group 2.

5.2.2.4 **(type)** – bracket group 2 – the word type.

5.2.2.5 **(?=\s*\=)** – **positive lookahead** – checking the condition after the highlighted groups.

5.2.2.6 The literal specified in the block: typeParam 2 ?<=@@simpad)\s(.*)(type)(?=\s*\=).

5.2.3 The **@BLOCKTYPE** block is a section for specifying text blocks. The section includes:

5.2.4 ID of the text block type.

5.2.5 **The type of block** is a literal consisting of an integer and an alphanumeric index.

5.2.6 Numerical code:

5.2.6.1 **1** is a simple block consisting of one literal from the list.

5.2.6.2 **2** is a block between two literals. The literal of the end of the block is at the END of the list. Other literals are possible literals of the beginning of the block. The literals of the beginning and end of the list are included in the block.

5.2.6.3 **3** is a block between two literals. The literal of the end of the block is at the BEGINNING of the list. Other literals are possible literals of the beginning of the block. The literals of the beginning and end of the list are included in the block.

5.2.6.4 **2x** is a block between two literals. The literal of the end of the block is at the END of the list. Other literals are possible literals of the beginning of the block.

5.2.6.5 **21** – The literals of the beginning and end of the list are included in the block.

5.2.6.6 **22** – The literals of the beginning are included in the block, the literal of the end of the list is not included in the block.

5.2.6.7 **23** – The literals of the beginning are not included in the block, the literal of the end of the list is included in the block.

5.2.6.8 **24** – The literals of the beginning and end of the list are not included in the block.

5.2.6.9 **3x** is a block between two literals. The literal of the end of the block is at the END of the list. Other literals are possible literals of the beginning of the block. The value of the second digit of the number is similar to the 2x types.

5.2.7 Alphanumeric index (may be missing. By default – A)

5.2.7.1 **A** – in the whole document

5.2.7.2 **B** – everywhere except @@simpad blocks

5.2.7.3 **D** – in the sections @@datafile and @@datafile_n

5.2.7.4 **Dn** – in sections @@datafile_n

5.2.8 List of literals – identifiers - links to the entry in the LITERAL section

5.2.9 Predefined text blocks:

5.2.9.1 **INLINE** – block of inline formulas

5.2.9.2 **_EDIT_** – editable macro variables

5.2.9.3 **_MATH_** – calculated macro variables

5.2.9.4 **_COMLINE_** – lowercase comments

5.2.9.5 **_COMBLOCK_** – comment block

5.2.9.6 **_COMPY_** – Python comments in math and func blocks

5.2.10 The **@STYLE** block is a section for setting the font style.

5.2.10.1 Style identifier.

5.2.10.2 The name of the font type, including **default or 0** (zero) – the default font type. The default value is 0.

5.2.10.3 Fat content – **B** – fat, **N** – usually (default)

5.2.10.4 Italics – **C** – italics, **N** – usually (default)

5.2.10.5 Font color – three components R G B – literal from the list of integers from 0 to 255 without spaces, or the name of the color according to CSS Color Level 4. If not specified, then the default color

5.2.10.6 Background color. If not specified, then the default color is

5.2.11 If there is an asterisk symbol instead of a value in any field, it means that this value is not set

5.2.12 The **@STATIC** block is a section for setting the permanent selection style in the entire document.

5.2.12.1 Priority is an integer. The higher the number, the higher the priority.

5.2.12.2 The style ID is a link to the style in the FONT section.

5.2.12.3 List of block type identifiers in the BLOCK section.

5.2.13 The **@DINAMIC** block is a section for setting the style of dynamic block selection at the cursor location.

5.2.13.1 Priority is an integer. The higher the number, the higher the priority.

5.2.13.2 The style ID is a reference to the style in the FONT section.

5.2.13.3 List of block type identifiers in the BLOCK section.

5.3 Lists of syntax highlighting styles.

5.3.1 For syntax highlighting, several styles can be specified in the form of an ordered list of styles. The styles set at the beginning of the list have a higher priority than the styles at the end of the list. That is, the first block found in the style files is highlighted in the order of their assignment in the list.

5.3.2 All identifiers specified in the style file are local, i.e. defined within a single style file.

5.4 Configuring syntax highlighting styles.

5.4.1 The style files are placed in the appropriate installation folder of the program.

5.4.2 The default syntax highlighting style is set in the configuration file, and a list of styles is provided for the specified file extensions. Styles are set by the name of the corresponding file.

6. Service files

6.1 The _property property file.

6.1.1 The _property property file is .A txt file that postprocessors create or add to with JSON data and/or images in the _IMG subfolder of the session folder.

6.1.2 The @@post command indicates the beginning of a set of data blocks for the postprocessor. Post processors use this data to create or supplement a file _property.txt with information in JSON format.

6.2 JSON data in the _property file:

6.2.1 The object created by the postprocessor. The mnemonic name of the object is set by the postprocessor. Each postprocessor creates its own object.

6.2.2 **Properties** object - contains objects with property values and their characteristics

6.2.2.1 **Property name** – a short designation (identifier) of the property

6.2.2.2 **vartype** – type of the property value

6.2.2.2.1 **“real"** is a real number (by default)

6.2.2.2.2 **“int"** is an integer

6.2.2.2.3 **“string"** is a string

6.2.2.2.4 **“ramp"** – ramp is a tabular function of two valid properties.

6.2.2.3 **value** – the value of the property. For the ramp, the spread is the difference between the maximum and minimum values

6.2.2.4 **tolerance** – the value of the absolute error of the property. It is used in simPad to round and represent numbers with the required number of significant digits. the default is 1 for integers and 0.1% for real numbers.

6.2.2.5 **quantity** – the identifier of the physical quantity in the Quantities object

6.2.2.6 **comment** – description of the property. For example, "The temperature of the upper zone in room 1"

6.2.2.6.1 **Language name** – an object with a property description string, for example, "ru": "Temperature of the upper zone in room 1"

6.2.2.7 **proptype** – property type – string literal without spaces. By default, it is missing, which means that this property is the value of some value without additional properties and parameters

6.2.2.8 **extra** – an object with additional properties and parameters, depending on the value of the proptype. it may be missing.

6.2.2.9 **ramp** – ramp values object

6.2.2.9.1 **value1** – array of ordinate values – in ascending order

6.2.2.9.2 **value2** – an array of abscissa values

6.2.2.9.3 **err1** is an array of ordinate error values. If it is missing, then the errors are 0

6.2.2.9.4 **err2** is an array of abscissa error values. If it is missing, then the errors are 0

6.2.2.9.5 **quantity2** – the identifier of the physical quantity of the ordinate in the Quantities object

6.2.2.10 **tag** is an array of integers – codes (tags) for grouping properties during processing - building tables, graphs, etc.

6.2.2.11 **tagstr** is a string tag – a string in which any information is encoded for grouping properties during processing. For example, "A01 B03"

6.2.3**Quantities** object – physical quantities

6.2.3.1 **The name of the value** is the short name (identifier) of the value.

6.2.3.1.1 **symbol** – the designation of a physical quantity, for example, "T".

6.2.3.1.2 **name** – the name of a physical quantity.

6.2.3.1.3 **Language name** – an object with a description string. For example, "ru": "Temperature"

6.2.3.1.4 **umane** is the name of a unit of physical quantity, for example, "degree Celsius"

6.2.3.1.5 **usymbol** - for example "C"

6.2.4the **Units** object – contains objects of units of physical quantities

6.2.4.1 **Unit name** – short name (identifier) of a unit of physical quantity

6.2.4.1.1 **quantity** – identifier of a physical quantity

6.2.4.1.2 **name** – the name of the unit of physical quantity,

6.2.4.1.3 **Language name** – an object with a description string. For example, "ru": "degree Celsius"

6.2.4.1.4 **symbol** – for example "C"

6.2.4.1.5 **factor** – multiplier – coefficient of the function of reducing the unit U to the base unit Ub. U=U * factor + offset. By default, factor = 1.0

6.2.4.1.6 **offset** – offset By default offset = 1.0

6.2.5The **Proc** object contains objects describing the processing procedure

6.2.5.1 **postprocessor** – postprocessor description string

6.2.5.2 **date** – date of processing

6.2.5.3 **time** – processing time

6.2.6the **ImageList** object is an array of arrays

6.2.6.1 **type** – number – 11- static bitmap, 12- static vector image 21-animated bitmap

6.2.6.2 **the name of the image file** contained in the _IMG subfolder of the session folder

6.2.6.3 description of the image