# VisualDOC

## LIST OF NEW FEATURES

***VERSION 7.2***

July 2013

1767 SOUTH 8TH STREET, SUITE 200
COLORADO SPRINGS, CO 80905
Phone: (719) 473-4611 Fax: (719) 473-4638
http://www.vrand.com
Email: visualdoc.support@vrand.com

## VisualDOC 7.2

VisualDOC 7.2 includes several major and minor changes aimed at improving the user-interface and performance as compared to version 7.1. The highlight of this release is batch processing of design points and the addition of parallel simulation capability to VisualDOC. A new component that communicates data with the Genesis software is also added. This new component does not require the user to manually specify how to read/write genesis data files. The following is the list of additions and enhancements in this release of VisualDOC.

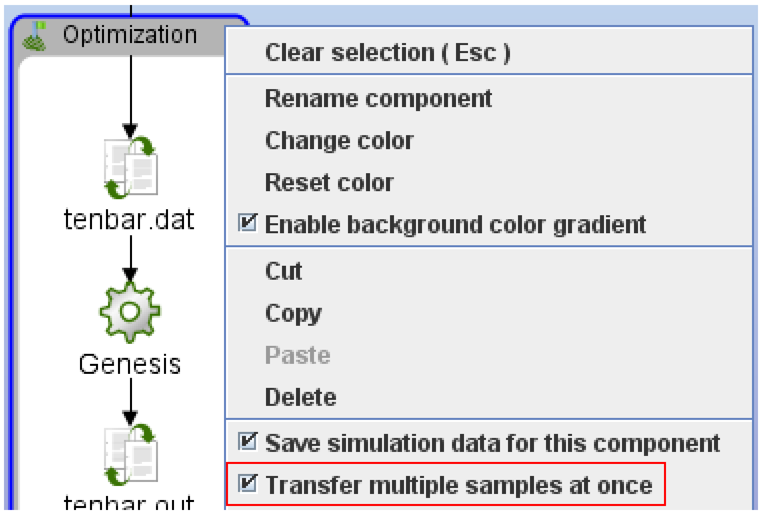
## Parallel Simulation Capability

VisualDOC 7.2 includes parallel simulation capability. All the design components (Optimization, Design of Experiments, Response Surface Approximation, and Probabilistic Analysis) and the For Loop component can potentially drive their sub-flow in parallel. The parallel simulation capability in VisualDOC is primarily intended for running the user's analysis program in parallel (launch multiple runs on local and remote computers simultaneously with different input data). In VisualDOC 7.2, a design component (e.g. Design of Experiments) can generate multiple sets of inputs (multiple samples) at once, each of which can be evaluated simultaneously by launching multiple instances of the user's analysis program with different input data.

Two sets of new capabilities have been added to VisualDOC. Both these capabilities help in speeding up the simulation process.

### Batch Processing

The first capability is batch processing of design points. Instead of generating one set of points (one sample), the design component and the *For* loop component can now generate more than one set of points at once (multiple samples). All the samples can be passed at once to the subflow. Each component in the subflow is now capable of processing a batch of points either sequentially or simultaneously. Transferring and processing data in batches reduces the total execution time. The batch processing of data for any eligible component can be enabled by selecting the checkbox "**Transfer multiple samples at once**" as shown in **Figure 1**.



☑ Enable background color gradient
☑ Save simulation data for this component
☑ Transfer multiple samples at once

**Figure 1  Multiple Sample Configuration UI**

## Parallel Execution

Certain components such as *Executable*, *Executable Wrapper*, and *Genesis* can process all the samples at once. These components can launch multiple runs of the analysis program with different input data at the same time and collect all the results for each sample to significantly reduce the total computation time. The total number of samples that can be processed at once depends upon the total number of available nodes. The parallel simulation configurator is shown in **Figure 2**.

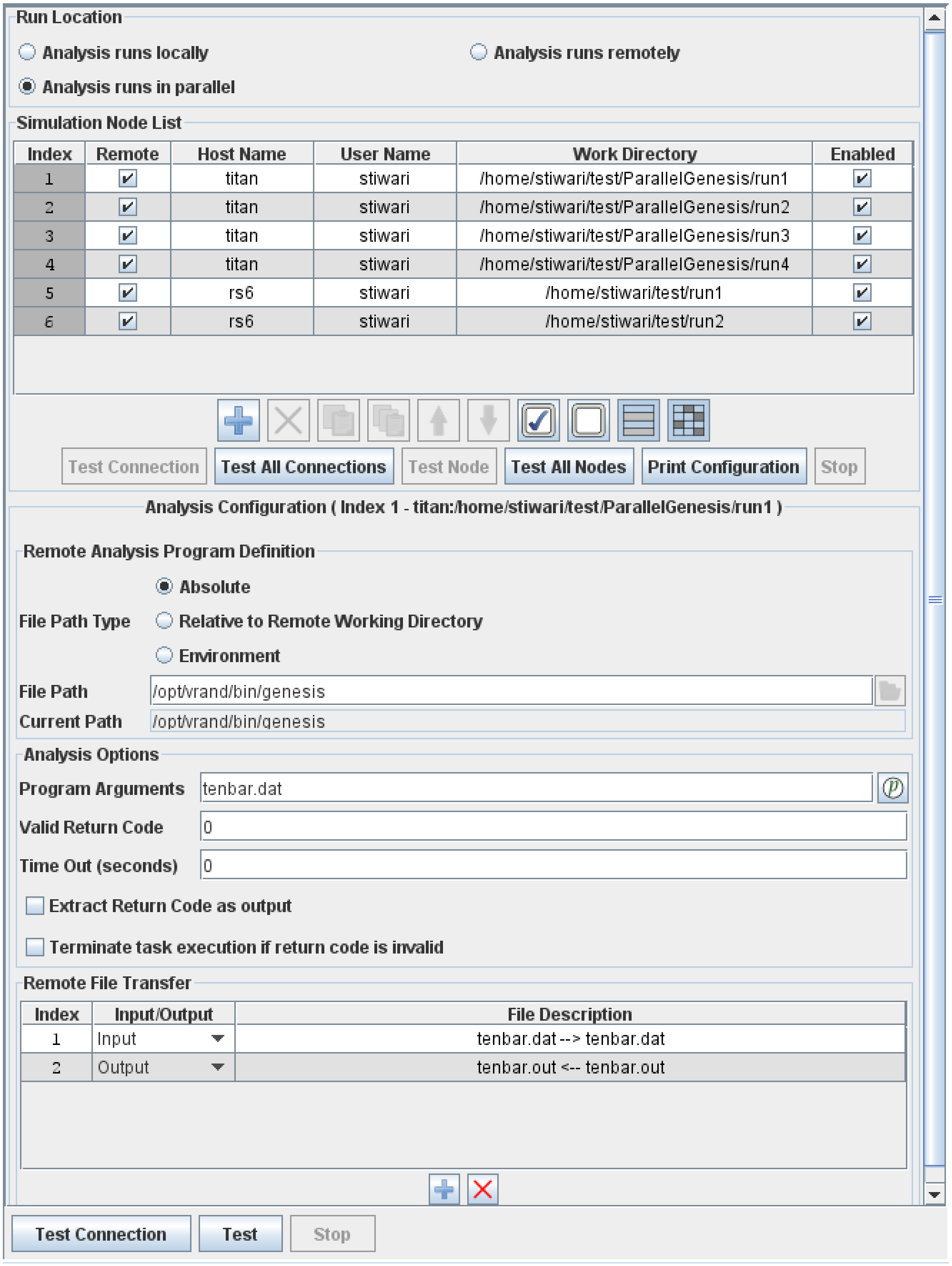
The following are the characteristics of the parallel simulation capability in VisualDOC.

1. The design/control components themselves do not run in parallel. They however generate multiple samples (sets of inputs) at any iteration. All the generated samples can then be potentially evaluated in parallel by an analysis component.
2. While Loop component can itself generate multiple samples at once but not drive a workflow in parallel. This is due to the conditional nature of its execution. The condition in the while loop (which may depend on previous iteration) determines whether the subflow should be run in the next iteration. Similarly, the If component can work with multiple samples but cannot branch in parallel (it either follows the True or the False branch at any given iteration).
3. All the analysis components are compatible with parallel simulation capability. All such components can accept, process, and return multiple samples at once.
4. It is assumed that the user's analysis program takes almost all the time during a real simulation, and therefore only the user's analysis is run in parallel during a simulation.
    4.1. Most VisualDOC components (except File I/O) take few milliseconds to run for typical data sizes and therefore parallelizing them is generally not required.
    4.2. The performance of the File I/O component depends upon the hard-disk (storage media) speed and is generally unaffected by the CPU speed. Parallelizing this component may actually reduce the performance (increase the total execution time) because of simultaneous I/O overhead. In general, updating the input files and extracting results from the output file still takes significantly less time than the user's analysis program in most scenarios.
5. The typical overhead of running an analysis in parallel (on remote computers) is 0.5 seconds per execution node. This overhead can be attributed to launching of scp/ssh commands and book-keeping done by VisualDOC. It is therefore recommended to not parallelize an analysis program that takes less than 0.5 seconds to run. An analysis program that takes few seconds (to few hours) to run is ideally suited for parallelization.

The following are the features of parallel simulation capability in VisualDOC.

1. VisualDOC only requires password-less ssh authentication to be setup between the client (the desktop on which VisualDOC is running) and the remote machines.
2. VisualDOC does not need/use MPI for parallel run. Hence, the user is not required to setup/configure MPI to run VisualDOC in parallel. VisualDOC 6.2.2 depended on MPI for parallel simulation capability.
3. VisualDOC does not need/use python on remote (or local) machines for parallel run. VisualDOC 6.2.2 required python to be installed on the remote machines for the parallel run.
4. The operating system on the remote machine can be anything (Windows, Linux, Unix, Solaris, etc.). The only requirement is the password-less ssh authentication and the ability of the user's analysis program to run on the remote computer.

5. No configuration files are needed. Entire setup (including validation/monitoring) is done through an easy-to-use GUI which is part of the property editor of the Executable (and Executable Wrapper) component. The parallel setup in VisualDOC is equivalent to configuring remote run for multiple different machines at the same time.
6. VisualDOC includes the ability to clone (copy/paste) the configuration of a node for multiple similar nodes. With this ability, the user can configure a single node, then create multiple clones, and edit each clone (e.g. change the hostname or the working directory) as desired to quickly setup parallel execution. VisualDOC also includes multi-clone feature to create multiple clones using a user-defined pattern for host name or the working directory.
7. Every configurable property can be independently modified for each node if desired. For example, the location of analysis program may vary from one machine to another. Also, the user may want to specify different command line arguments for different nodes (e.g. to vary the number of threads used by the analysis program on a specific machine).
8. For each node, the user can specify the number of runs to perform on that machine. For example, if a machine has multiple cores/processors, the user can choose to perform multiple simultaneous runs on that machine. In such a case, the user is required to specify separate working directories for each run.
9. The user can prioritize the order in which the nodes should be used (the faster nodes can be specified before the slower nodes). VisualDOC assigns jobs to the available nodes in the order they are specified by the user. If multiple nodes are available at any time, the faster node will always be used first. VisualDOC itself does not have any mechanism to determine which nodes are faster. It is therefore recommended that the user specify the faster node before a slower node for optimum performance.
10. The user can enable/disable a node. A disabled node is simply not used (ignored) for job submission.
11. VisualDOC 7.2 supports asynchronous job submission. It is possible that a faster node may be assigned more jobs than a slower node. As soon as a node finishes the assigned job, it is included in the pool for remaining jobs and is chosen in the user-specified order.
12. A facility is added to print the entire configuration in textual format for easy review and validation. The user can also independently check connection and check execution of each (or all) node(s) at once.



Run Location

( ) Analysis runs locally
( ) Analysis runs remotely
(●) Analysis runs in parallel

Simulation Node List

| Index | Remote | Host Name | User Name | Work Directory | Enabled |
|---|---|---|---|---|---|
| 1 | ✓ | titan | stiwari | /home/stiwari/test/ParallelGenesis/run1 | ✓ |
| 2 | ✓ | titan | stiwari | /home/stiwari/test/ParallelGenesis/run2 | ✓ |
| 3 | ✓ | titan | stiwari | /home/stiwari/test/ParallelGenesis/run3 | ✓ |
| 4 | ✓ | titan | stiwari | /home/stiwari/test/ParallelGenesis/run4 | ✓ |
| 5 | ✓ | rs6 | stiwari | /home/stiwari/test/run1 | ✓ |
| 6 | ✓ | rs6 | stiwari | /home/stiwari/test/run2 | ✓ |

Analysis Configuration ( Index 1 - titan:/home/stiwari/test/ParallelGenesis/run1 )

Remote Analysis Program Definition

File Path Type: (●) Absolute ( ) Relative to Remote Working Directory ( ) Environment

File Path: /opt/vrand/bin/genesis

Current Path: /opt/vrand/bin/genesis

Analysis Options

Program Arguments: tenbar.dat

Valid Return Code: 0

Time Out (seconds): 0

☐ Extract Return Code as output

☐ Terminate task execution if return code is invalid

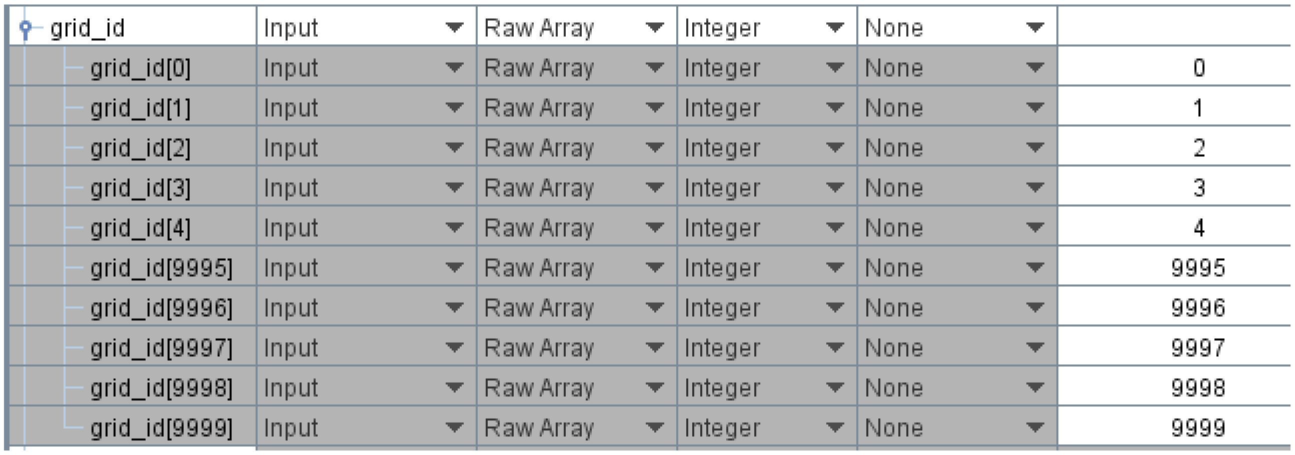
Remote File Transfer

| Index | Input/Output | File Description |
|---|---|---|
| 1 | Input | tenbar.dat --> tenbar.dat |
| 2 | Output | tenbar.out <-- tenbar.out |

**Figure 2 Parallel Simulation Property Editor**

## Raw Integer Array

VisualDOC 7.1 added a new data type namely “Raw Array” to support large data up to few million in size. The data type only supported real values. VisualDOC 7.2 adds support for integers to Raw Array data. The user can now use Raw Arrays to represent integer values. To use the integer data type, the user can simply choose “integer” as value type in the data editor. A Raw Array data with integer values is shown in **Figure 3**.

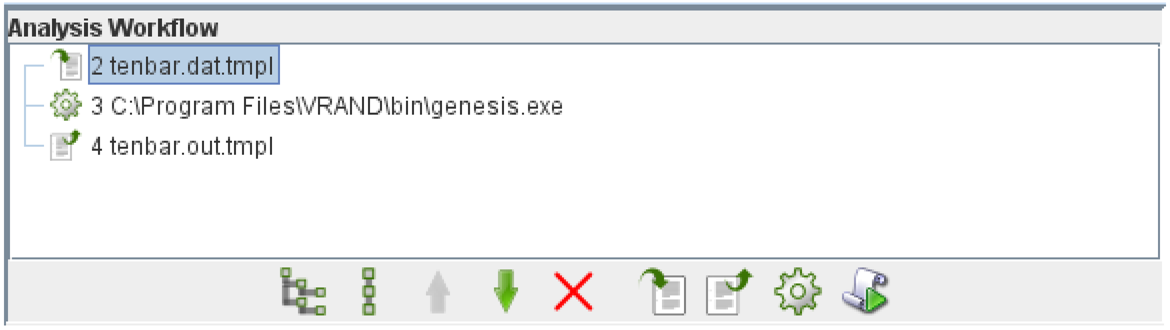


| | | | | | |
|---|---|---|---|---|---|
| grid_id | Input | Raw Array | Integer | None | |
| grid_id[0] | Input | Raw Array | Integer | None | 0 |
| grid_id[1] | Input | Raw Array | Integer | None | 1 |
| grid_id[2] | Input | Raw Array | Integer | None | 2 |
| grid_id[3] | Input | Raw Array | Integer | None | 3 |
| grid_id[4] | Input | Raw Array | Integer | None | 4 |
| grid_id[9995] | Input | Raw Array | Integer | None | 9995 |
| grid_id[9996] | Input | Raw Array | Integer | None | 9996 |
| grid_id[9997] | Input | Raw Array | Integer | None | 9997 |
| grid_id[9998] | Input | Raw Array | Integer | None | 9998 |
| grid_id[9999] | Input | Raw Array | Integer | None | 9999 |

**Figure 3 Raw Array data with integer values**

## New Genesis Component

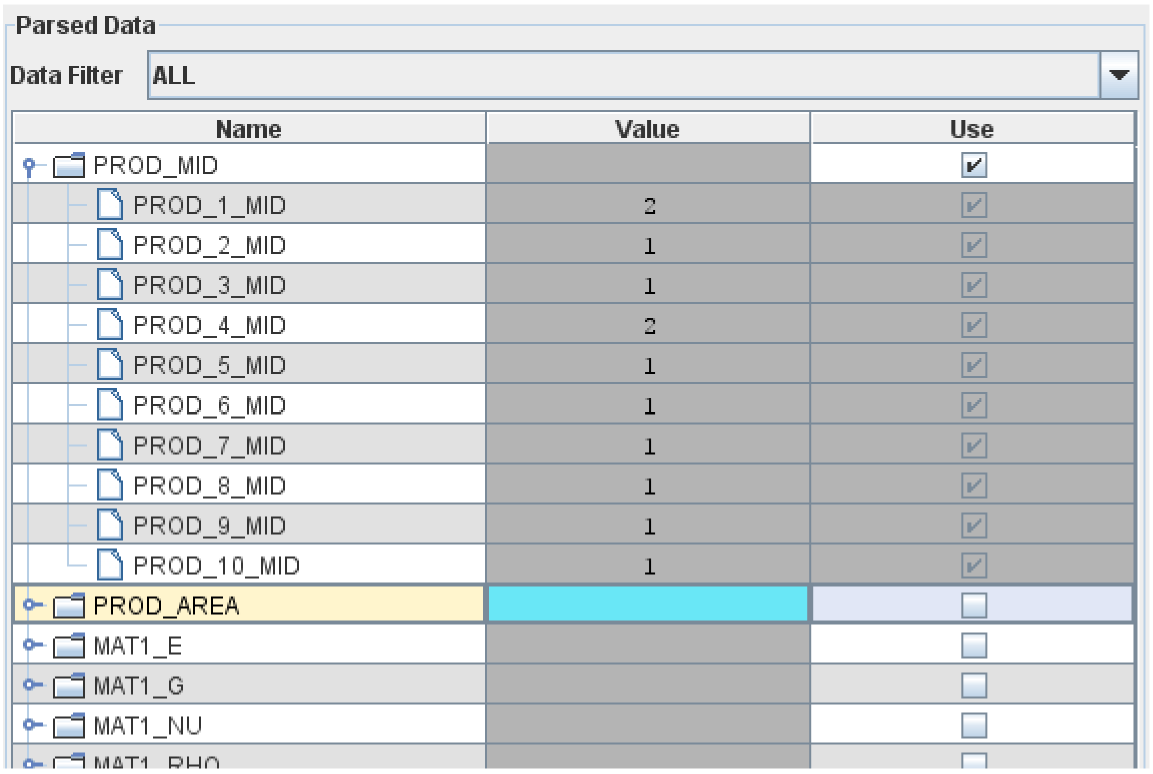
The new Genesis component is an integrated file IO component that talks directly to Genesis. It allows the user build an analysis workflow in this single component. The workflow (**Figure 4**) typically incudes Genesis input file (*.dat file), Genesis executable, and Genesis output file (*.out or punch file).



2 tenbar.dat.tmpl
3 C:\Program Files\VRAND\bin\genesis.exe
4 tenbar.out.tmpl

**Figure 4 Genesis Component Workflow**

Compared with File I/O component, the user does not need to create search/modify/extract actions by working with the text files, which is tedious and error prone in general. Instead, Genesis component automatically parses and shows the supported type of input data in a tree table as shown in **Figure 5**. The user can also easily add the outputs to be extracted using the provided data filters as shown in **Figure 6**. The “Use” checkbox allows the corresponding data be added/removed from the data treetable.

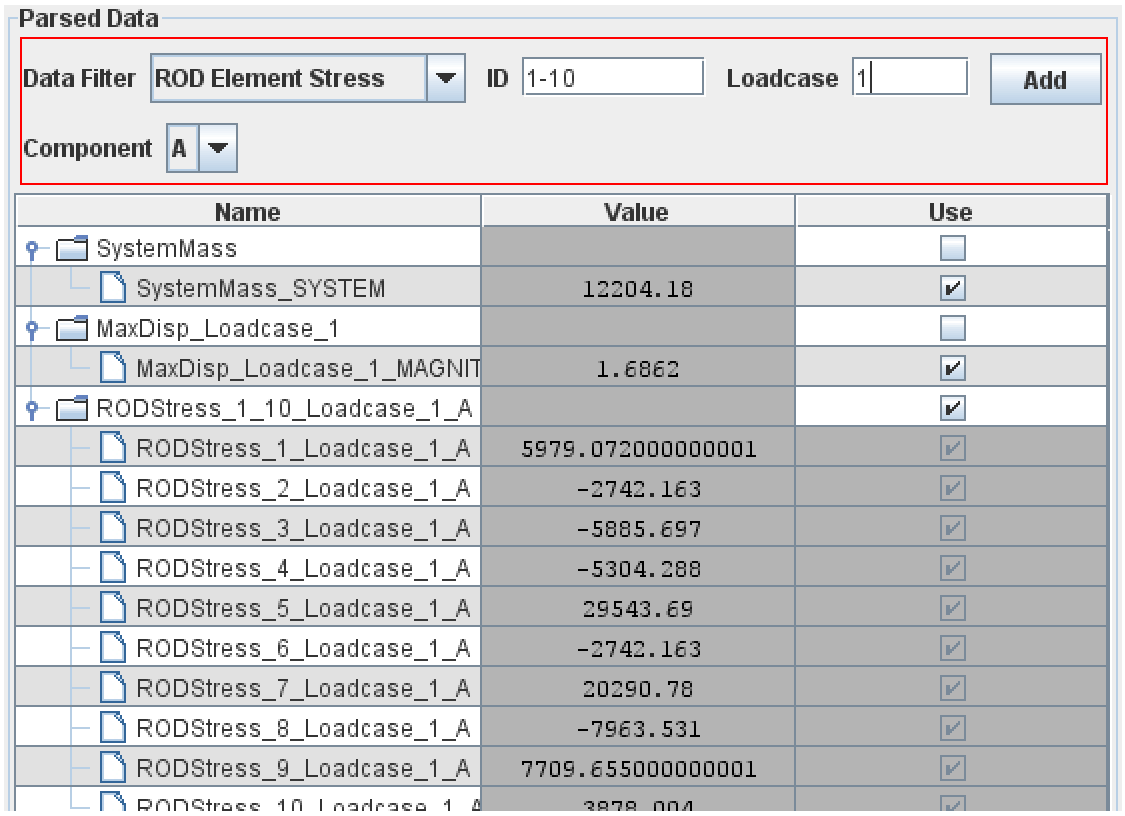


Parsed Data

Data Filter: ALL

| Name | Value | Use |
|---|---|---|
| PROD_MID | | ✓ |
| PROD_1_MID | 2 | ✓ |
| PROD_2_MID | 1 | ✓ |
| PROD_3_MID | 1 | ✓ |
| PROD_4_MID | 2 | ✓ |
| PROD_5_MID | 1 | ✓ |
| PROD_6_MID | 1 | ✓ |
| PROD_7_MID | 1 | ✓ |
| PROD_8_MID | 1 | ✓ |
| PROD_9_MID | 1 | ✓ |
| PROD_10_MID | 1 | ✓ |
| PROD_AREA | | ☐ |
| MAT1_E | | ☐ |
| MAT1_G | | ☐ |
| MAT1_NU | | ☐ |
| MAT1_RHO | | ☐ |

**Figure 5 Parsed Data for Genesis Input File**



Parsed Data

Data Filter: ROD Element Stress    ID: 1-10    Loadcase: 1    Add

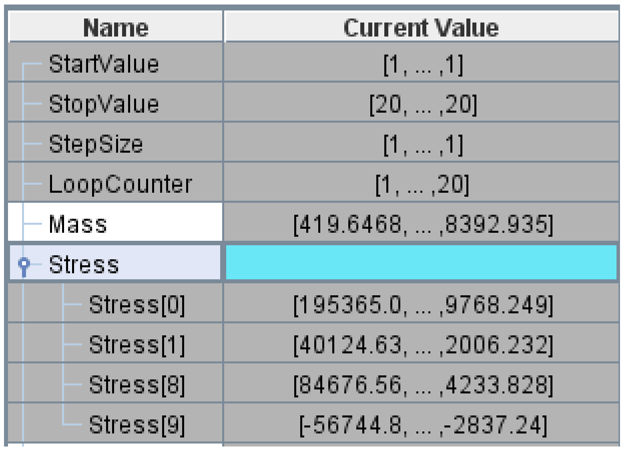
Component: A

| Name | Value | Use |
|---|---|---|
| SystemMass | | ☐ |
| SystemMass_SYSTEM | 12204.18 | ✓ |
| MaxDisp_Loadcase_1 | | ☐ |
| MaxDisp_Loadcase_1_MAGNIT | 1.6862 | ✓ |
| RODStress_1_10_Loadcase_1_A | | ✓ |
| RODStress_1_Loadcase_1_A | 5979.072000000001 | ✓ |
| RODStress_2_Loadcase_1_A | -2742.163 | ✓ |
| RODStress_3_Loadcase_1_A | -5885.697 | ✓ |
| RODStress_4_Loadcase_1_A | -5304.288 | ✓ |
| RODStress_5_Loadcase_1_A | 29543.69 | ✓ |
| RODStress_6_Loadcase_1_A | -2742.163 | ✓ |
| RODStress_7_Loadcase_1_A | 20290.78 | ✓ |
| RODStress_8_Loadcase_1_A | -7963.531 | ✓ |
| RODStress_9_Loadcase_1_A | 7709.655000000001 | ✓ |
| RODStress_10_Loadcase_1_A | 3878.004 | ✓ |

**Figure 6 Parsed Data for Genesis Output File**

## Multiple Sample Viewer

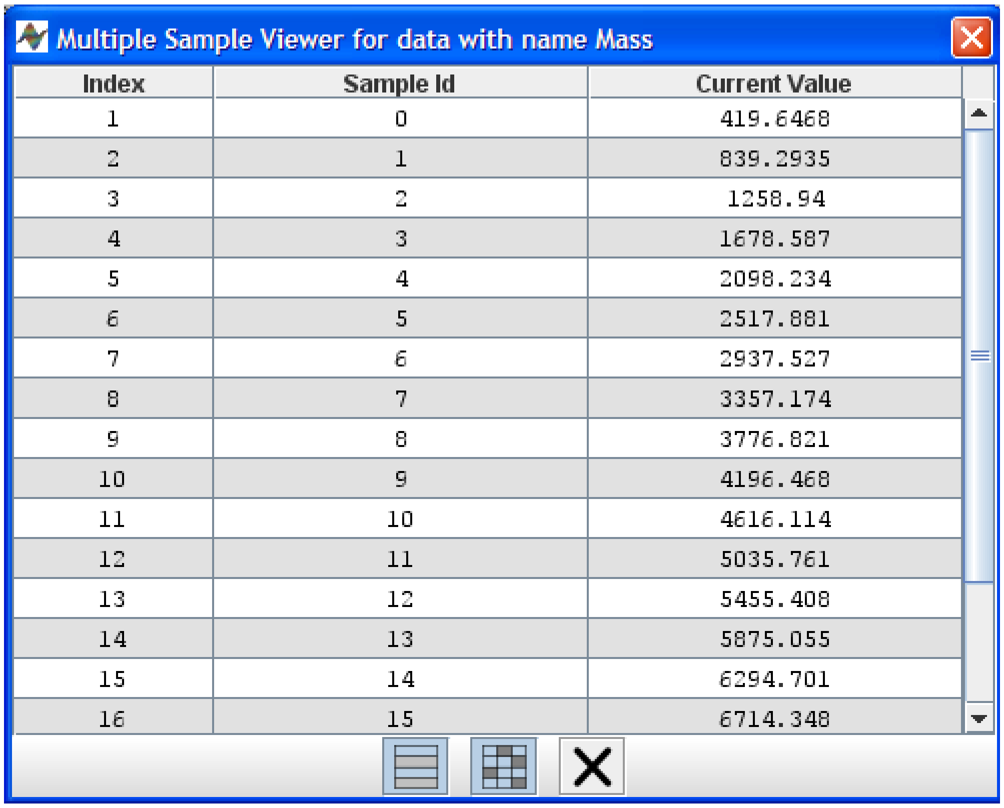
VisualDOC 7.2 can generate (and use) multiple values for each data at once. To display multiple values at once, a multiple sample viewer is included. The usual tables and tree-tables (throughout VisualDOC) display multiple samples as shown in **Figure 7**.



| Name | Current Value |
|---|---|
| StartValue | [1, ... ,1] |
| StopValue | [20, ... ,20] |
| StepSize | [1, ... ,1] |
| LoopCounter | [1, ... ,20] |
| Mass | [419.6468, ... ,8392.935] |
| Stress | |
| Stress[0] | [195365.0, ... ,9768.249] |
| Stress[1] | [40124.63, ... ,2006.232] |
| Stress[8] | [84676.56, ... ,4233.828] |
| Stress[9] | [-56744.8, ... ,-2837.24] |

**Figure 7  Multiple Sample Display in a Table**

The notation in **Figure 7** only shows the value of the first and the last sample. Whenever more than one values are present, VisualDOC use the notation in **Figure 7** to display the values in any UI. If the user wishes to examine the values of all the samples, then Multiple Sample Viewer can be launched by choosing it from the context dependent popup menu. The Multiple Sample Viewer is shown in **Figure 8**. This viewer shows the *Current Value* of any data for all the samples.



Multiple Sample Viewer for data with name Mass

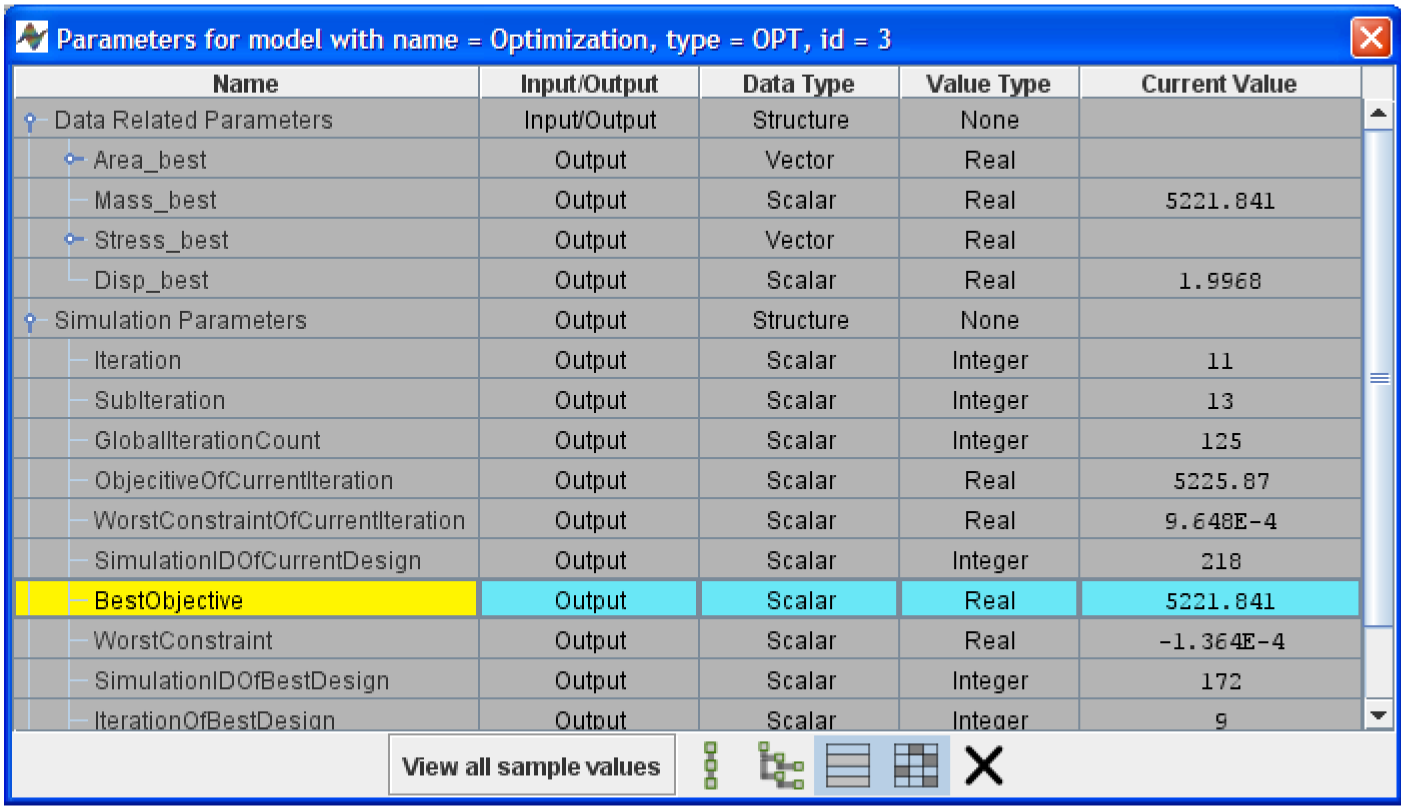
| Index | Sample Id | Current Value |
|---|---|---|
| 1 | 0 | 419.6468 |
| 2 | 1 | 839.2935 |
| 3 | 2 | 1258.94 |
| 4 | 3 | 1678.587 |
| 5 | 4 | 2098.234 |
| 6 | 5 | 2517.881 |
| 7 | 6 | 2937.527 |
| 8 | 7 | 3357.174 |
| 9 | 8 | 3776.821 |
| 10 | 9 | 4196.468 |
| 11 | 10 | 4616.114 |
| 12 | 11 | 5035.761 |
| 13 | 12 | 5455.408 |
| 14 | 13 | 5875.055 |
| 15 | 14 | 6294.701 |
| 16 | 15 | 6714.348 |

**Figure 8  Multiple Sample Viewer**

## Parameter Viewer

Parameters (data that are not created/edited by the user) in VisualDOC are automatically created and configured by different components. Since, user does not create or configure such data, no UI existed to monitor their values. With Version 7.2 a new *Parameter Viewer* has been added that can show the values of all parameters contained in a component. The parameter viewer is shown in **Figure 9**. The following are some of the features of the parameter viewer.

1. It can be used as a monitor in that when a simulation is running, the parameter values are automatically updated as the simulation proceeds.
2. Any number of parameter viewers can be open at the same time. The user can open a viewer for each component if desired.
3. For the case of parallel run, the parameter viewer can launch multi-sample viewer for any parameter that has multiple values.
4. The viewer can be opened and operated while the model is being edited. The viewer automatically closes if the corresponding model is deleted.
5. The viewer cannot be used to edit the parameters since the parameters by definition are not user-editable.
6. The viewers automatically update when new parameters are added or existing parameters are deleted.


Parameters for model with name = Optimization, type = OPT, id = 3
View all sample values

| Name | Input/Output | Data Type | Value Type | Current Value |
|---|---|---|---|---|
| Data Related Parameters | Input/Output | Structure | None | |
| Area_best | Output | Vector | Real | |
| Mass_best | Output | Scalar | Real | 5221.841 |
| Stress_best | Output | Vector | Real | |
| Disp_best | Output | Scalar | Real | 1.9968 |
| Simulation Parameters | Output | Structure | None | |
| Iteration | Output | Scalar | Integer | 11 |
| SubIteration | Output | Scalar | Integer | 13 |
| GlobalIterationCount | Output | Scalar | Integer | 125 |
| ObjecitiveOfCurrentIteration | Output | Scalar | Real | 5225.87 |
| WorstConstraintOfCurrentIteration | Output | Scalar | Real | 9.648E-4 |
| SimulationIDOfCurrentDesign | Output | Scalar | Integer | 218 |
| BestObjective | Output | Scalar | Real | 5221.841 |
| WorstConstraint | Output | Scalar | Real | -1.364E-4 |
| SimulationIDOfBestDesign | Output | Scalar | Integer | 172 |
| IterationOfBestDesign | Output | Scalar | Integer | 9 |

**Figure 9 Parameter Viewer**

## New Message Component ()

A new component to print a message during execution is added. This component does not have a property editor and cannot contain data. The sole purpose of this component is to print status messages. It prints a single line message every time it is executed. Thus, this component can also be used as a filler component (when a workflow does not need any component but must have one for semantic validity).

## Other Modifications

- The display of component names in the VisualDOC workflow has been modified. If the name of a component contains more than a specified number of characters, the font size is reduced so that the entire name can be displayed. VisualDOC continues to reduce the font size up to a specified limit so as to display the entire name. If the name is too long and still cannot not be displayed completely, then it is trimmed appropriately.
- An option to copy the "*Optimum values*" to the "*Initial values*" before running the Optimization component has been added. By default, this check-box is disabled (the values are not copied). The user can select this check-box if it is desired to start the optimization using the best found value (in a previous run) as the starting point. The user can also choose to not copy the best values in the first run, but copy it in subsequent runs.
- DOT/BIGDOT Parameter Dialog - The DOT/BIGDOT "Additional Parameters Dialog" allows the user modify additional control parameters of DOT/BIGDOT optimizer that are not shown in the general property section. This gives users the full flexibility of tuning the optimizer for their particular design problem.
- BIGDOT 4.0 - Version 4 of BIGDOT contains two significant enhancements over earlier versions. One is algorithmic and the second provides more user control. The algorithmic change is in the way internal multipliers are calculated. The new method is more efficient in terms of iterations to achieve an optimum, but takes significantly more computational time than the original method. Therefore, the new method is recommended for problems where there are expected to be less than about 2000 active constraints. The user control option now allows the user three levels of default parameters. With this, the user can choose to achieve a near optimum very quickly or can choose to continue the optimization process to achieve a more precise optimum.
- In version 7.1, when changing data type from "Vector" to "Raw Array", the user was required to confirm all the attributes of data to complete the task. In version 7.2, changing between compatible "Vectors" and "Raw Arrays" does not require the user to confirm the other attributes. The attributes are simply copied and modified appropriately during the switch.
- All the bugs discovered since version 7.1 have been fixed in this release.

## Version Compatibility

One of the features of VisualDOC is to be able to communicate with various different software such as Matlab and Excel.Each release of VisualDOC is tested with the earliest versions that it supports and also the latest release of different software products. VisualDOC has been tested with the following software versions.

- Java

VisualDOC works with both the 32- and 64-bit Java. Java versions from 6.x to the version 7 update 21 are supported. Java 7 is recommended. VisualDOC will not run on Java version 4 or earlier. Support for version 5 is deprecated and some of the functionality may not work reliably in Java 5.

- Windows

  VisualDOC works on both the 32- and 64-bit Windows. It has been tested on Windows XP, Vista, 7, and 8. On Windows 8, VisualDOC defaults to the classic UI and does not support the new Metro interface.

- Linux

  Only the 64-bit Linux distributions that have kernel version 2.6.x are supported. VisualDOC works with most major distributions of Linux.

- Microsoft Excel

  Excel versions 1997 to 2010 are supported. It is strongly recommended to use Excel 2003 or later with VisualDOC.

- Matlab

  Release 2009a to Release 2013a are supported. Matlab releases prior to 2009a are not supported.

Even though VisualDOC supports older versions of the software that it integrates with, it is strongly recommended to use the latest versions of the software whenever possible. It is especially suggested to use Java 7 or later for performance and efficiency reasons.

## New Examples and Documentation

Seven new examples showcasing the new and advanced features of VisualDOC have been added with this release. The new examples are as follows.

1. MDO-MDF Formulation: This simple example showcases how to solve a MDO problem using the MDF formulation.
2. MDO-IDF Formulation: This simple example showcases how to solve a MDO problem using the IDF formulation.
3. MDO-CO Formulation: This simple example showcases how to solve a MDO problem using the CO formulation. It also shows how to do book-keeping with VisualDOC for multi-level optimization.
4. ParallelTenBarTruss: The Ten Bar Truss example has been modified to run in parallel. The executable program and the source code for the analysis is included with the example. The user can run this parallel example without the need of any external software.
5. ParallelGenesis: The Genesis interface example has been modified to run in parallel. Genesis software must be available to the user to run this example. Also, the user must have multiple Genesis licenses to be able to run this example in parallel.
6. tenbar_mat1_selection: This example showcases how to use Genesis component to modify the material id of properties. It also shows how to generate a set of candidate designs in DOE and select the best design according to user defined criteria.
7. tenbar_mat1_topology: This example showcases how to use Genesis component to modify the material properties and do a topology optimization.

All the accompanying manuals have been updated for this new release.