

OASIS User Manual

May 02, 2022

Warranty and Software Agreement

This Trial Software Licence (the “**Agreement**”) is made between EMPOWER OPERATIONS CORP., a Canadian corporation with its principal place of business in Surrey, British Columbia, Canada (“**Empower**”), and individuals or entities (the “**Customer**”) that acquire a right to use the Software (as defined below). This Agreement establishes the terms under which Empower will license the Software to Customer.

By clicking on the applicable “I agree” button, downloading, copying, installing, activating or otherwise using the Software, Customer does so with the intent to electronically “execute” and agree to be bound by this Agreement. IF CUSTOMER DOES NOT AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT, Customer has no right to use the Software and Customer should return, delete or disable the Software.

1. Definitions

In this Agreement, unless expressed to the contrary:

“*Confidential Information*” means all information designated in writing as confidential by each party, or which under the circumstances of disclosure reasonably ought to be considered as confidential or the nature of which is such that it would generally be considered confidential in the industry in which the disclosing party operates. Without limiting the foregoing, Empower’s Confidential Information also includes the Software, including all source and object code, Documentation, and the terms and conditions of this Agreement.

“*Device*” means the single, specifically identified computing device or virtual machine on which Customer installs the Software.

“*Documentation*” means the Empower-created and -supplied user and system administrator guides, codes and manuals relating to the Software.

“*Software*” means a trial version of the software marketed by Empower as “Optimization Assisted System Integration Software” (OASIS).

“*Trial Period*” means a period of thirty (30) days, measured from the exact time and date of the initial installation of the Software, or other durations as determined by Empower.

“*User(s)*” means any employee, agent, contractor or other representative of Customer or person granted access by Customer who uses the Software under Customer’s Trial Licence (as defined herein).

Other terms may be defined elsewhere in the text of this Agreement and will have such meaning throughout this Agreement.

2. Licence Grant

- a. Subject to the terms of this Agreement, Empower grants to Customer for the Trial Period and at no cost to Customer a non-exclusive, non-transferable, royalty-free licence to: (i) install and use the Software on a Device; (ii) use the Documentation in conjunction with the use of the Software; and (iii) allowing the Users to exercise the foregoing rights (collectively, the “**Trial Licence**”).
- b. The Trial Licence is to be used by Customer solely for purposes of evaluating the Software for purchase. The Trial Licence is for use of the machine-readable object code only, excluding any source code.
- c. Upon the expiry of the Trial Period, the Software will automatically cease operation, including any processes or simulations being run by the Software at such time. If Customer wishes to continue to use the Software (including accessing or using any data saved by Customer or additional modules available for the Software) after the expiry of the Trial Period, Customer will need to (i) save any work in progress prior to the expiry of the Trial Period; and (ii) purchase a licence to use the Software from Empower. The Customer acknowledges and agrees that it will not be able to access or use any data saved by Customer using the Software during the Trial Period if it fails to purchase a licence to use the Software from Empower after expiration of the Trial Period.

3. Customer Acknowledgements and Obligations

- a. Customer will install, use and operate the Software in accordance with the Documentation and any further recommendations and requirements specified by Empower.
- b. Customer acknowledges and agrees that: (i) the Software will automatically create and store log files (“**Log Files**”) that are related to use of the Software on the relevant Device; and (ii) Log Files may contain Customer’s Confidential Information or other sensitive information about Customer’s business inputted by Users into the Software. Log Files will not be accessible to Empower unless access to Log Files is provided to Empower by Customer.
- c. If Customer provides Empower with access to Log Files, in association with delivery of support to Customer or otherwise, Customer is solely responsible for removing or redacting any Confidential Information or other sensitive business information contained in Log Files prior to providing any Log Files to Empower.
- d. Customer is solely responsible for the purchase or licensing of all equipment and software (other than the Software)

- necessary to install and properly operate the Software as detailed in the Documentation.
- e. Customer will be responsible for all acts or omissions of the User contrary to the terms and conditions of this Agreement.

4. Restrictions on Use

- a. Customer will not install the Software on any hardware, equipment or computing device other than the Device.
- b. Customer may only install the Software on the Device once using a Trial Licence.
- c. Customer will not request, download, install or use an additional copy of the Software using a different name or username.
- d. Customer will not use the Software for any commercial purpose, including the development of products or services, while operating under a Trial Licence.
- e. Customer will not, and will ensure that Users do not: (i) decompile, disassemble, reverse engineer or modify the object code of the Software; (ii) access or attempt to access any modules or other portions of the Software that are disabled or otherwise not included in the Trial Licence; or (iii) use the Software to develop products that compete with the Software.
- f. Except as provided in Section 12f, Customer is prohibited from distributing, transferring possession of, or otherwise making available the Software or Documentation to any other person (other than to Users).

5. Ownership and Copies

- a. All right, title and interest, including all intellectual property rights, in and to: (i) the Software and Documentation; (ii) the media on which the Software and Documentation are provided to Customer; and (iii) any ideas or know-how developed by Empower through the provision of support or other services to Customer belong exclusively to Empower or its licensors. Customer acknowledges that, except as specifically provided under this Agreement, no such right, title or interest in these items is transferred or otherwise granted by the Trial Licence.
- b. Customer will not at any time, whether before or after the termination of this Agreement, contest or aid others in contesting or doing anything that otherwise impairs the validity of any proprietary and intellectual property rights, title or interest of Empower in and to the Software or Documentation.
- c. Customer is permitted to make: (i) copies of the Documentation for each User; and (ii) one (1) copy of the Software for back-up or archival purposes. Such authorized copies of the Software will contain all proprietary markings or legends specified by Empower.

6. Confidentiality

- a. Except as reasonably required to exercise its rights under this Agreement, Customer agrees to prevent any unauthorized copying, use, distribution, installation or transfer of possession of Empower's Confidential Information. At a minimum, Customer will maintain at least the same procedures regarding Empower's Confidential Information that it maintains with respect to its own Confidential Information of a similar type. Customer will not acquire any interest in any Confidential Information received from Empower by reason of this Agreement. Nothing in this Agreement will restrict Empower's use or disclosure of its own Confidential Information. Confidential Information does not include any information that: (i) becomes part of the public domain through no act or omission of Customer; (ii) is lawfully acquired by Customer from a third party without any breach of confidentiality; (iii) is independently developed without reference to the Confidential Information of Empower; or (iv) is disclosed in accordance with judicial or other governmental order or timely disclosure requirements imposed by law or stock exchange policies. Notwithstanding the foregoing, Customer will be permitted to disclose the terms and conditions of this Agreement when required by law or regulation. Without limiting the generality of the foregoing, Customer will take reasonable steps to prevent any personnel or User from removing any proprietary or other legend or restrictive notice contained or included in any material provided by Empower to Customer.
- b. Customer acknowledges that any use or disclosure of Empower's Confidential Information in a manner inconsistent with the provisions of this Agreement may cause Empower irreparable damage for which remedies other than injunctive relief may be inadequate. Customer further agrees that Empower will be entitled to attempt to receive from a court of competent jurisdiction injunctive or other equitable relief to restrain such use or disclosure in addition to other appropriate remedies.
- c. The obligations under this Section 6 will apply to employees, contractors and agents of Customer and any others, including any Users, using the Software under Customer's Trial Licence.

7. No Warranty

The Software and Documentation are supplied by Empower on an "AS-IS" basis. Upon Customer's request, Empower may provide support for the Software during the Trial Period on such terms and conditions as may be agreed upon by the parties.

8. WARRANTY DISCLAIMER

TO THE EXTENT PERMITTED BY LAW, EMPOWER DISCLAIMS, AND CUSTOMER WAIVES, ALL REPRESENTATIONS, WARRANTIES OR CONDITIONS, WHETHER EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING ANY WARRANTY OR CONDITION OF MERCHANTABILITY, MERCHANTABLE QUALITY, DURABILITY, TITLE, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE SOFTWARE, DOCUMENTATION AND ANY OTHER ITEMS OR SERVICES PROVIDED UNDER THIS AGREEMENT AND NOTHING IN THIS AGREEMENT IMPLIES ANY WARRANTY THAT THE OPERATION OF

SOFTWARE WILL BE UNINTERRUPTED OR ERROR FREE, OR THAT ANY ERRORS FOUND WILL BE CORRECTED.

9. Indemnity

Customer will indemnify, defend and hold harmless Empower and its directors, officers, employees and agents from and against any third-party claims, actions, demands, loss, liability (including amounts paid in settlement) or costs or expenses (including legal fees on a solicitor and client basis) arising out of or in connection with: (a) any modifications, changes, adaptations, extensions or uses of the Software by Customer or its Users that infringe, violate or misappropriate any proprietary or other right of any third party, including any intellectual property rights; and (b) any breach by Customer of this Agreement, including breach of Section 6.

10. Limitation of Remedies

- a. To the maximum extent permitted by law, Empower will not be responsible for any loss or damage to Customer or any third parties caused by the Software or by Empower's performance under this Agreement.
- b. TO THE MAXIMUM EXTENT PERMITTED BY LAW, EMPOWER, ON BEHALF OF ITSELF AND ITS SUPPLIERS, DISCLAIMS ANY AND ALL LIABILITY FOR (I) SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES (INCLUDING LOSS OF PROFITS); AND (II) PUNITIVE AND EXEMPLARY DAMAGES ARISING OUT OF THIS AGREEMENT OR WITH RESPECT TO OR IN CONNECTION WITH THE INSTALLATION, IMPLEMENTATION, USE, OPERATION, OR SUPPORT OF THE SOFTWARE AND DOCUMENTATION, EVEN IF EMPOWER OR ITS SUPPLIERS HAVE BEEN APPRISED OF THE POSSIBILITY OF SUCH DAMAGES.
- c. THE LIMITATIONS OF THIS SECTION 10 WILL APPLY TO ALL CAUSES OF ACTION, WHETHER BASED ON BREACH OF WARRANTY, BREACH OF CONDITION, BREACH OF CONTRACT, FUNDAMENTAL BREACH OR BREACHES, INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, NEGLIGENCE, OTHER TORT CLAIMS, STRICT LIABILITY OR ANY OTHER LEGAL OR EQUITABLE THEORY.
- d. THIS SECTION WILL SURVIVE TERMINATION AND WILL APPLY NOTWITHSTANDING THE FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY.
- e. CUSTOMER ACKNOWLEDGES AND AGREES THAT THE DISCLAIMERS, EXCLUSIONS AND LIMITATIONS SET FORTH IN THIS AGREEMENT CONSTITUTE AN ESSENTIAL ELEMENT OF THIS AGREEMENT IN THE ABSENCE OF WHICH: (A) THE TERMS OF THIS AGREEMENT WOULD BE SUBSTANTIALLY DIFFERENT; AND (B) EMPOWER'S ABILITY TO OFFER AND CUSTOMER'S ABILITY TO ACCESS AND USE THE SOFTWARE UNDER THIS AGREEMENT WOULD BE IMPAIRED.

11. Termination

- a. This Agreement will terminate automatically upon the expiry of the Trial Period, unless terminated earlier in accordance with this Agreement.
- b. Empower may for any reason, and without liability or obligation to Customer, terminate this Agreement by providing notice in writing to Customer.
- c. Upon termination of this Agreement, regardless of the cause, any licence, including the Trial Licence, granted under this Agreement is immediately revoked. Within ten (10) business days after such termination, Customer will: (i) return to Empower or delete and destroy all copies of the Software and Documentation in Customer's possession; and (ii) upon request by Empower, deliver a certificate of an officer of Customer certifying that the same has been completed. Termination of this Agreement will be in addition to, and not in lieu of, any other remedies available to either party.

12. Audit Rights

During the Trial Period and for a period of twelve (12) months after expiry of the Trial Period, Empower, at its own expense, may audit Customer's use of the Software and Customer's compliance with the terms of this Agreement. Any such audit will be conducted during regular business hours at Customer's facilities and will not unreasonably interfere with Customer's business activities. If an audit reveals that Customer has failed to pay any required licence fees due to Empower, Customer will be invoiced for such unpaid fees at Empower's then-current list prices. If the Customer is found to have materially breached this Agreement, Customer will also pay Empower's reasonable costs of conducting the audit.

13. Miscellaneous

- a. Each party acknowledges that it has read and understands this Agreement and further agrees that it is the complete and exclusive statement of the agreement between the parties that supersedes and merges all prior proposals, understandings, and all other agreements, oral and written, between the parties relating to the subject matter of this Agreement.
- b. This Agreement may not be modified or altered except by written instrument duly executed by both parties. Any terms and conditions of any purchase order or other instrument issued by Customer in connection with this Agreement that are in addition to or inconsistent with the terms and conditions of this Agreement are not binding on Empower.

- c. Any notice or other communication required or permitted by this Agreement will be in writing and will be deemed to have been duly given on the day of service if served personally or by facsimile transmission with confirmation, the day of receipt by the recipient if served via e-mail, or three (3) days after mailing if mailed by registered or certified mail, postage prepaid, and addressed to the respective parties at their respective corporate headquarters.
- d. This Agreement and performance under this Agreement will be governed by the laws of the Province of British Columbia, Canada, and the federal laws of Canada applicable therein, without regard to conflicts of law principles that would apply a different body of law. The United Nations Convention on Contracts for the International Sale of Goods and any local implementation of such convention, including the International Sale of Goods Act of British Columbia, will not apply in any way to this Agreement or to the transactions contemplated by this Agreement or otherwise to create any rights or to impose any duties or obligations on any party to this Agreement. The parties irrevocably submit to and accept generally and unconditionally the exclusive jurisdiction of the courts of the province of British Columbia with respect to any legal action or proceeding that may be brought at any time relating in any way to this Agreement.
- e. If any provision of this Agreement is invalid under any applicable statute or rule of law, it is to that extent to be deemed omitted. The remainder of this Agreement will be valid and enforceable to the maximum extent possible.
- f. Customer may not assign or sub-license, without the prior written consent of Empower, any of its rights, duties, or obligations under this Agreement to any person or entity, in whole or in part.
- g. The waiver or failure of either party to exercise in any respect any right provided for in this Agreement will not be deemed a waiver of any further right under this Agreement.
- h. Both parties agree to comply with all export and re-export restrictions and regulations imposed by the government of Canada or the United States, or corresponding or similar laws of other countries where Customer is using the Software.
- i. Nothing in this Agreement will be construed to create an agency, joint venture, partnership, or other relationship between the parties. No agent, employee, or representative of either party has the authority to bind the other party in any manner. The parties are independent contractors with respect to each other under this Agreement.
- j. Neither party will be responsible for failure to perform in a timely manner under this Agreement when its failure results from any of the following causes: Acts of God or public enemies, civil war, insurrection or riot, fire, flood, explosion, earthquake or serious accident, strike, labour trouble or work interruption or any cause beyond its reasonable control. This section will not apply to excuse any failure to make any payment when due.
- k. The terms of Sections 1, 4a, 4b, 4c, 4d, 5, 6, 7, 8, 9, 10, 11c, 12 and 13 will survive termination of this Agreement.
- l. Subject to the restrictions on transfer contained in this Agreement, this Agreement will enure to the benefit of and be binding on the parties and their respective successors and assigns.

Contents

Getting Started	1
1 Introduction	2
2 System Requirements.....	4
2.1 Visual C++ 2017 Redistributable.....	4
Problem Definition.....	5
3 Parametric Problem Setup.....	6
3.1 Variables.....	6
3.1.1 Creating Variables	6
3.1.2 Allowed Variable Names.....	7
3.1.3 Setting Variable Properties	7
3.2 Outputs	9
3.2.1 Babel Expressions.....	10
3.2.2 PowerShell Expressions.....	12
3.2.3 Simulation Outputs	15
3.3 Simulation Integration.....	15
Simulation Integration	18
4 General Simulation Integration	19
4.1 Simulation Integration Edit Window	19
4.2 Executable Configuration.....	20
4.3 Exit Codes	22
4.4 Iterative Backup	23
4.5 Execution Strategy	24
4.6 Command-Line Options.....	25
4.7 Simulation Parameter Capture	26

4.7.1	Simulation Parameter Creation	26
4.7.2	Simulation Parameter Capture Steps	27
4.7.3	Detection Window	28
4.8	Simulation Integration MATLAB Example.....	30
4.9	Command Line Options Configuration.....	31
4.10	File Watcher SI Execution Strategy	32
4.10.1	Configuring Your Simulation Integration to Run with File Watcher Strategy.....	32
4.10.2	File Watcher Example.....	33
5	Multi-Simulation Configuration.....	35
6	PowerShell Integration.....	37
6.1	PowerShell Simulation Integrations.....	37
6.1.1	PowerShell Setup.....	37
6.1.2	Creating Custom PowerShell Scripts.....	37
6.1.3	Simulation Integration PowerShell Example.....	38
6.1.4	Using PowerShell with a Remote Machine or an HPC Cluster.....	40
7	Symbol Mapping.....	41
7.1	Symbol Mapping States	41
7.2	Create Mapping	42
7.3	Remove Mapping	44
	Optimization and Parametric Analyses.....	45
8	Design Verification	46
8.1	Verify Expression.....	46
8.2	Simulation Verification.....	47
9	Optimization.....	51
9.1	Optimizer.....	51

9.2	Import Points to Current Configuration	52
9.3	Export Optimization Data to Decision Making.....	52
10	Design of Experiments.....	54
10.1	DOE algorithms:	54
10.1.1	Latin Hypercube Design (LHD)	54
10.1.2	Full Factorial Design (FFD)	54
10.2	Generate and Evaluate a DOE Sampling Plan:	55
	Postprocessing.....	57
11	Results and Visualization.....	58
11.1	Parallel Coordinates Plot.....	58
11.1.1	Colored Point Bands	59
11.1.2	PCP Scrolling and Zooming	59
11.1.3	Fitness at the Right.....	60
11.2	Surface Scatter Plot (Single-Objective/ Multi-Objective)	60
11.2.1	Selection of Points	61
11.2.2	Detailed Point Information	62
11.2.3	Single-Objective Problem.....	62
11.2.4	Multi-Objective Problem	63
11.2.5	Pareto Frontier	63
11.3	Convergence Line Plot.....	64
11.4	Full Screen Visualization	66
11.5	Performance Panel.....	67
11.6	Excel Output of Results.....	68
11.7	Rerunning Points	68
11.8	End Report for Optimizations	69

12	Decision Making	70
12.1	Glossary.....	70
12.1.1	Attribute	70
12.1.2	Data.....	70
12.1.3	Rating.....	71
12.1.4	Sensitivity	71
12.2	Tools.....	71
12.2.1	Screening phase	71
12.2.2	Proceed Options	74
12.2.3	Result phase	76
12.3	Workflow	79
12.4	Case Study: Choosing Between a Software Product.....	79
	OASIS Operation and Navigation	83
13	Project Structure and Navigation.....	84
13.1	Project Files.....	84
13.2	Project History and Navigation Overview	85
14	OASIS Command Line	87
14.1	Setting up a Project for Execution in CLI Mode.....	87
14.2	Running a Project File in Command Line.....	87
14.2.1	OASIS CLI Options & Arguments Table.....	88
14.2.2	Example Calls	88
14.3	Viewing the Results	89
15	Troubleshooting	90
15.1	General Troubleshooting Tips.....	90
15.2	Logging	91

Getting Started

1 Introduction

OASIS is designed to help engineers find optimal product solutions based on given design variables and objectives. OASIS is driven by the following key features listed below.

- **Simulation Integration Support:** OASIS may optimize systems without mathematically defined input-output relationships (see Section 7). We can intelligently use simulations to model system responses mathematically while collecting sample-data (known as meta-modelling). As a result, OASIS can optimize a wide range of simulation models. The user simply needs to provide a path to the external simulation tool (e.g. ANSYS (see Section Error! Reference source not found.), MATLAB (see Section 4.8), etc.) and all the relevant simulation files (see Section 4).
- **Economic Use of Simulations:** Our algorithms are tailored to solve problems for good solutions while using *minimal* number of objective function evaluations. This means OASIS can optimize problems with large numbers of variables.
- **Flexible Problem Configuration (see Section 3):** We automatically select the best algorithms and parameters to match with your optimization problems. All the user needs to do is define variables, objectives and constraints, and OASIS will pick the right tools to solve your problem. There are no algorithm parameters to select or tune.
- **High Dimensional Visualization System (see Section 11):** OASIS provides clear and concise insight into your problem and optimization results. We feature an interactive parallel-coordinates graph that provides not only a visual summary of the data collected on optimization completion but also during live optimizations. Our choice of visualizations illustrates the effect of each variable on a problem's output. OASIS also contains a 3D visualizer that lets users interact with the search space in detail. Our convergence graph shows the optimization progress efficiently.
- **Design of Experiments (see Section 10):** Design of Experiments is a fundamental feature of in the parametric analysis toolbox. OASIS offers the design of experiments which will allow users to generate sample plans.
- **Portable and User-Modifiable Configuration Files (see Section Error! Reference source not found.):** Users can create easily modifiable configuration files that are

imported into OASIS. This serves as a hassle-free alternative to manually setting up problem definitions in OASIS. These configuration files are also portable and will work with varying versions our software.

- **Smart Simulation Backup (see 4.4):** During simulation runs OASIS intelligently creates backups of the working directory and all its contents at each evaluation. This optional feature not only serves as an easy-to-use debugging method for simulation runs but also allows users to make use of scripts that interacts with the working directory during an evaluation while the simulation is running.

2 System Requirements

Component	Minimum	Recommended
Operating System	Windows Vista and above	Windows 7 and above
Memory	2GB	4GB and above
Hard Drive	500MB of free space	8GB of free space ¹
Graphics	Intel 2 nd gen, AMD/ATI HD2000 Series, GeForce 8000 series	

2.1 Visual C++ 2017 Redistributable

OASIS requires an existing installation of the Visual C++ 2017 redistributable. The OASIS installer will attempt to install this component and will succeed in the installation of OASIS even if the installation of this particular component fails.

If you are encountering issues with the Visual C++ 2017 Redistributable (“vcruntime140.dll”), please attempt the following steps:

1. Make sure Windows updates are fully up-to-date. Note that the update KB2999226, “Universal C runtime for windows” is a prerequisite for the Visual C++ redistributable and thus for OASIS. See <https://support.microsoft.com/en-ca/help/2999226/update-for-universal-c-runtime-in-windows>
2. Download and run “vc_redist_x86” for visual studio 2017. See <https://support.microsoft.com/en-ca/help/2977003/the-latest-supported-visual-c-downloads>

If the KB2999226 patch is already installed and the Visual C++ 2017 Redistributable is also installed, and OASIS is still encountering errors about “vcruntime140.dll”, please contact support.

¹ Recommended free hard disk space depends on several factors such as the free disk space required by the operating system, other software such as simulation software .etc.

Problem Definition

3 Parametric Problem Setup

You can view the overall problem setup through the panel on the left, which also gives you quick access to the parameter to view and modify.

3.1 Variables

3.1.1 Creating Variables

Create a variable by typing its name in the *Name* text box, then click *Add*. If you want to create multiple variables with common names (e.g., x1, x2, x3), type the name into the *Name* text box and the number of desired variables in the adjacent box.

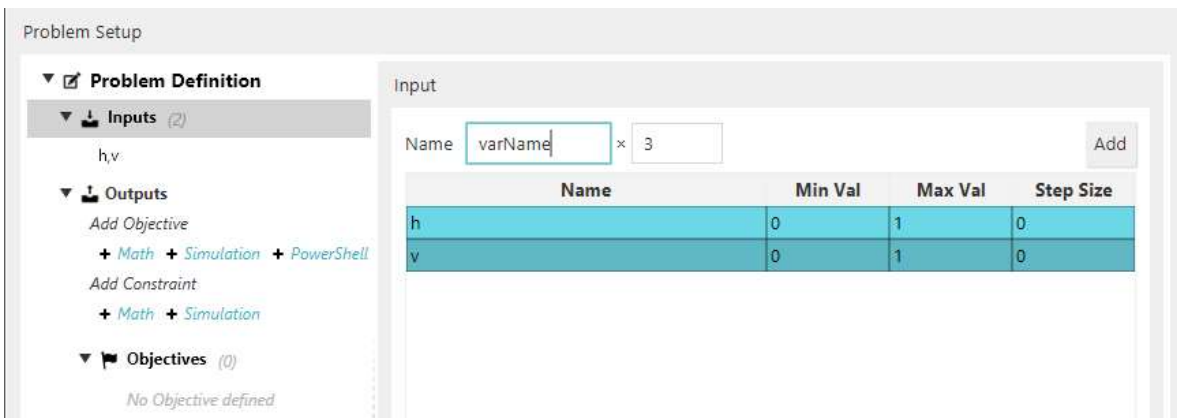


Figure 1: Variable Creation

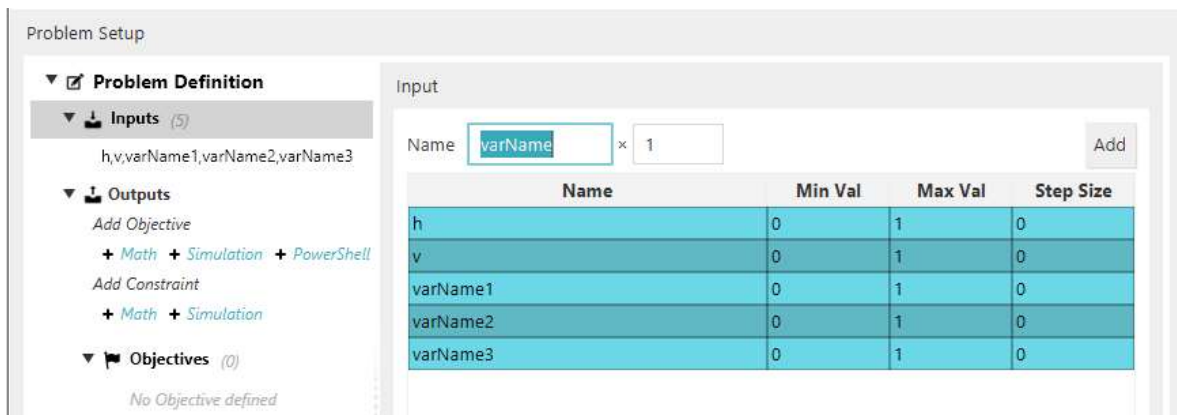


Figure 2: Multiple Variable Creation

3.1.2 Allowed Variable Names

OASIS variables are the aliases given to the optimization's simulation parameters. These aliases can be called many different things but must adhere to a few rules. Variable names:

- can use characters 'A' through 'Z' and 'a' through 'z'
- can use '_' (underscores).
- can use almost any Unicode character (such as σ , 变 and 量)
- can contain numbers
- cannot *start* with a number

For example, the following are legal variable names in OASIS:

- x1
- x_1
- σ
- _ σ
- 变量 1
- str_len_1234
- ALLCAPS_VARIABLE

The following are *not* legal variable names in OASIS:

- 1abc – starts with a number
- .abc – contains a dot

3.1.3 Setting Variable Properties

Bounds must be set for each variable. Multiple variables can be modified simultaneously when more than one is selected by SHIFT or CTRL clicking them (Figure 3). Variables can also be deleted by selecting them and clicking the *Delete* button.

Input

Name ×

Name	Min Val	Max Val	Step Size
h	0	1	0
v	0	1	0
VarName1	5	10	0
VarName2	5	10	0
VarName3	0	1	0

Min Max Step Variable Type

Figure 3: Multiple Selection and Modification of Variables

Variables can be *Continuous* or *Discrete*. By default, variables are continuous. When variables are set to discrete using the *Variable Type* drop-down menu, the step size also needs to be configured. Note that OASIS generates discrete values concerning the lower bound value, which means that depending on the step size, the largest feasible value for the variable may be smaller than the upper bound. For example, if the bounds are [-1, 1] with step size of 0.3, then the feasible values are -1, -0.7, -0.4, -0.1, 0.2, 0.5 and 0.8.

Input

Name ×

Name	Min Val	Max Val	Step Size
discreteVar1	0	1	.5
discreteVar2	0	1,000	200

Min Max Step Variable Type

Figure 4: Discrete Variable Configuration

3.2 Outputs

Outputs consist of *Objectives*, *Intermediates* and *Constraints*:

- *Objectives* are what OASIS optimizes by strategically exploring feasible solutions.
- *Intermediates* are a type of output that are used as *Inputs* for other *Objectives* or *Constraints*.
- *Constraints* are a type of output that define the feasibility of design points.

In simple problems, objective and constraint outputs defined in OASIS depend only on input variables. In some cases however, intermediate outputs need to be used for calculating the final objective outputs or constraints of an optimization. For example, if mathematical transformations (e.g., absolute value or penalty function) need to be applied to the outputs of an ANSYS Mechanical model, then the intermediate outputs would be captured from ANSYS Mechanical and the transformations would be applied to calculate the final objective outputs. Optimization problems require at least one objective to be considered valid. In visualizations, intermediate objectives can be identified by their surrounding angle brackets (< >). Intermediate *Constraints* are not supported at this time. All *Outputs* are in the form of either Math expressions, PowerShell expressions or simulation outputs. You add *Outputs* by clicking on the options available under *Outputs* in the *Problem Definition* section or via the pane on the right visible after clicking on *Outputs*. To define an output and edit its name, click on it and the edit pane will appear to the right.

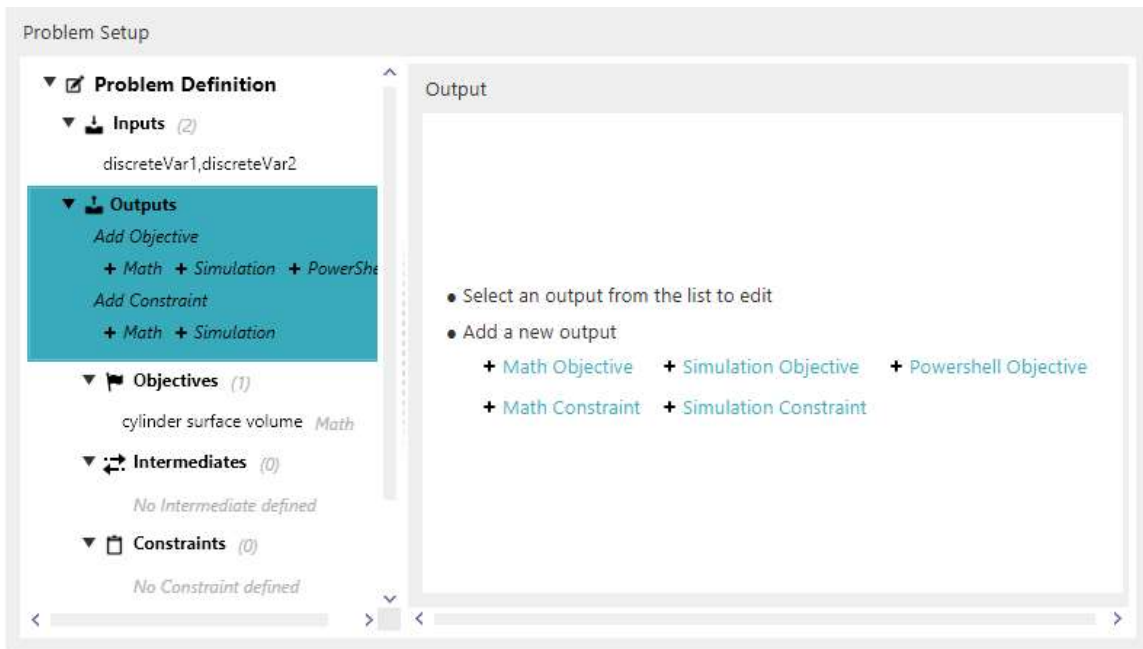


Figure 5: Output Creation in Problem Setup

3.2.1 Babel Expressions

Babel expressions are the preferred way of defining mathematical expressions. The language features syntax highlighting and is designed to be easy to use. For more information on Babel, see Section **Error! Reference source not found.** To create a Babel objective function, under the *Add Objective* header, select the *+Math* button. The default in OASIS is to minimize objective functions. To maximize the objective, just add a negative sign in front of the equation like the expression shown below for the *cylinder surface volume*.

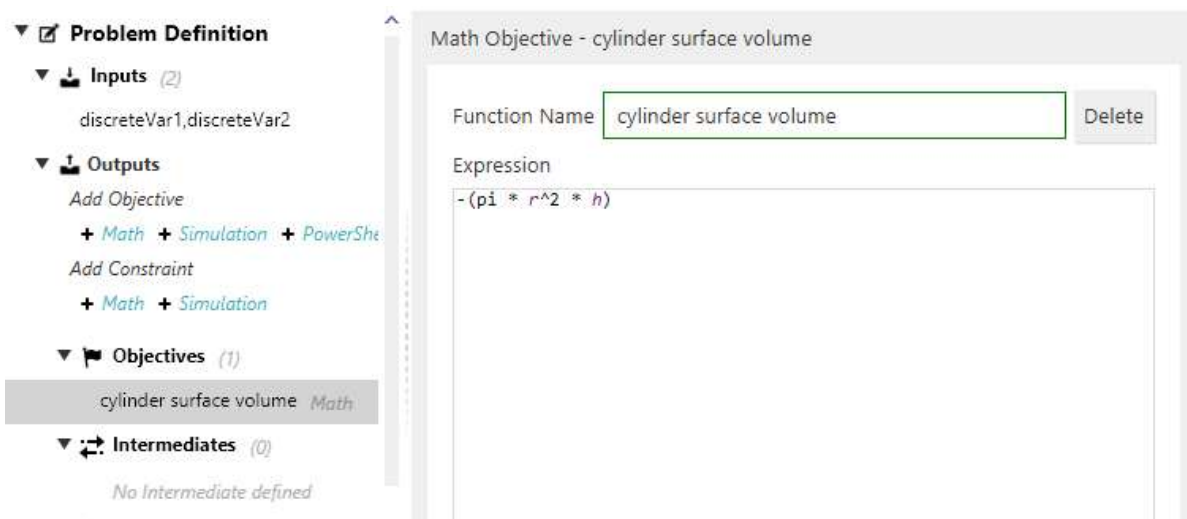


Figure 6: Objective Function Definition

If desired, *Constraint Functions* may also be added to the problem setup using *+Math* or *+Simulation* under the *Add Constraint* section of *Problem Definition*.

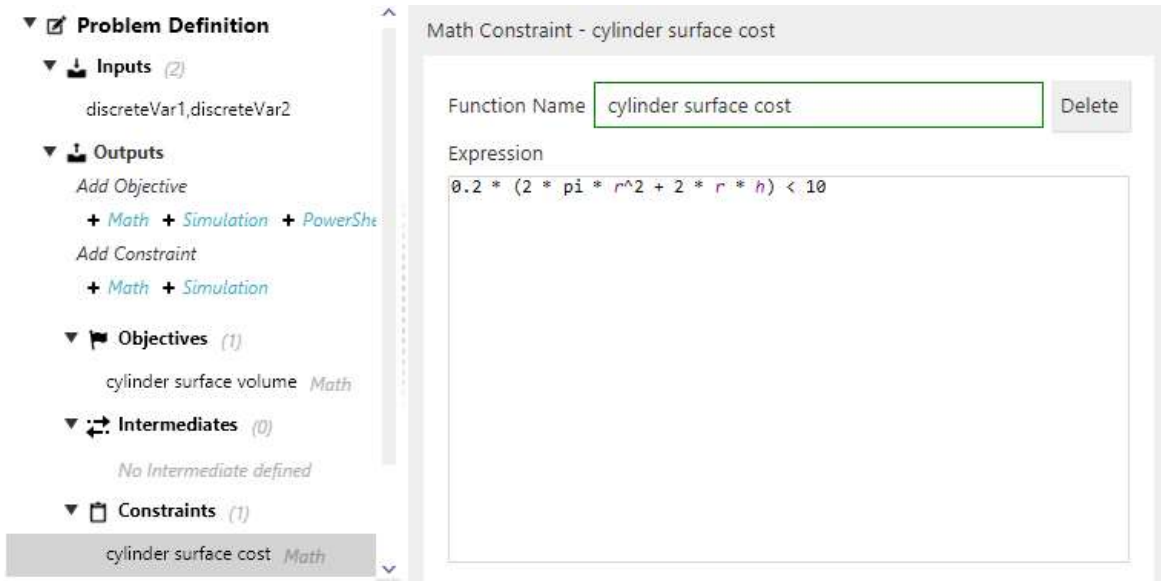


Figure 7: Constraint Function Definition

Babel offers several inequality operators to allow you to express constraints as relatively simple inequalities. These operators are listed in Section **Error! Reference source not found.**

Simulation constraints are more complex; when a Simulation that is declared as a constraint is evaluated in OASIS by assuming that negative values or zero represent constraint satisfaction, and positive values represent constraint failure.

Intermediate outputs can be added the same way as regular objective outputs. To make an objective output intermediate, simply reference it in any Babel expression. A referenced objective will automatically be labeled as an intermediate output.

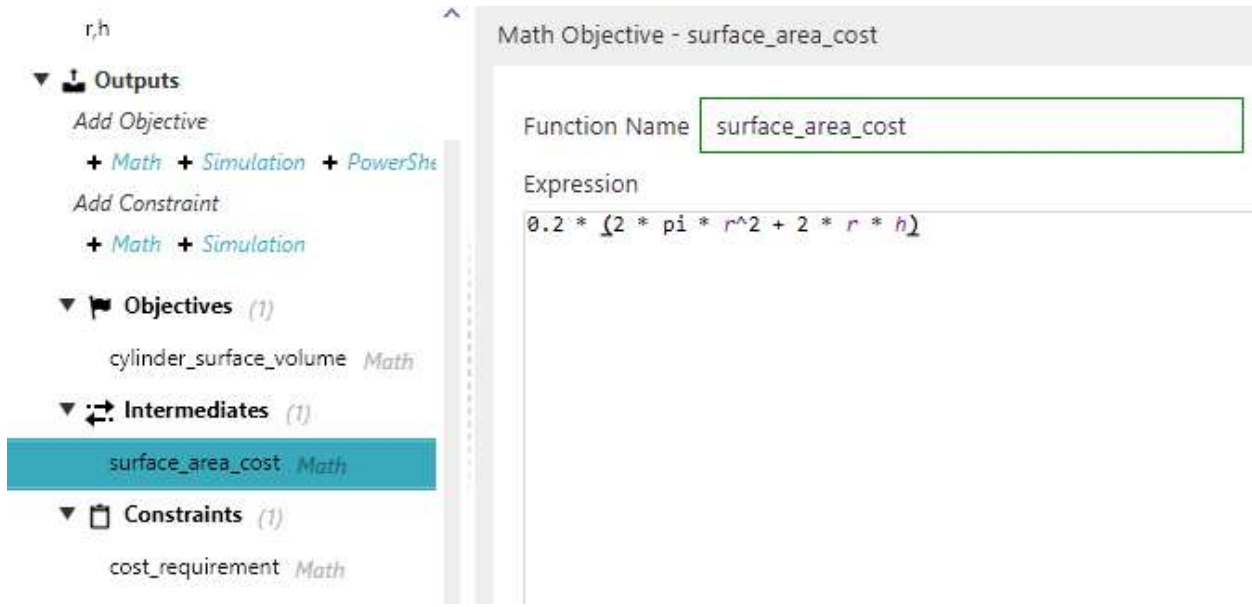


Figure 8: Defined an Intermediate Output

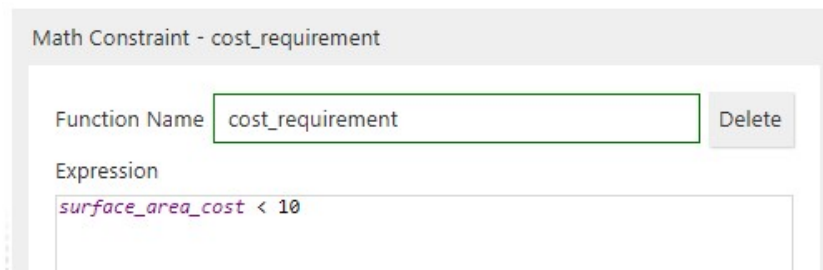


Figure 9: Constraint Expression using an Intermediate Output

3.2.2 PowerShell Expressions

PowerShell expressions are used in OASIS for complex and typically, for multi-line expressions that cannot be formulated as Babel expressions. Below each PowerShell expression there is a list with check-boxes that show whether the expression references existing objectives and/or constraints. To use PowerShell expressions, PowerShell must be set up and configured correctly on your machine. See Section 6.1.1 for setup and configuration information.

To create a *PowerShell* expression, select + *PowerShell* under *Outputs* in the *Problem Definition* section. **Note:** *PowerShell Constraints are currently not supported.*

Enter the *PowerShell* objective function in the expression field. The expression field contains a block of expressions or statements to be evaluated. Statements are non-value-producing operations such as variable assignments ($\$var1 = 2.0$), and expressions are operations that return

a value such as ``$var1``, ``$var2 + $var3``, or a plain value literal such as ``4.2``. The result of the PowerShell expression is the return value of the last operation in the block. A single-line expression of the form ``$x1 + $x2 + $x3`` will suffice. More complex multi-line expressions that involve intermediate steps will also work if the last operation returns the result. Hence, you can also write:

```
$temp1 = $x1 + $x2 # <- Statement.  
$temp2 = $x3 * 1.0 # <- Statement.  
$temp1 + $temp2   # <- Expression that returns the result.
```

This can also be used with control flow blocks:

```
if ($x -gt 0.5) {# <- Block begins.  
    $out = $x * 1.0 # <- Statement.  
} else {  
    $out = $x + 1.0 # <- Statement.  
} # <- Block ends. This block does not produce a value.  
$out # <- Expression that returns the result.
```

In the last example, you may also use expressions throughout:

```
if ($x -gt 0.5) {# <- Block begins.  
    $x * 1.0 # <- Expression.  
} else {  
    $x + 1.0 # <- Expression.  
} # <- Block ends. This block is now also an expression itself.
```

Both branches are now value-producing expressions, so the entire block is a value-producing expression itself and evaluates to either branch's result depending on which branch is taken. This last example can be found in *SC multi expression example.opyl* under the *Samples* folder.

The screenshot shows a configuration window titled "PowerShell Objective - f1". It contains a "Function Name" field with the value "f1" and a "Delete" button. Below this is a "PowerShell Expression" field containing the text "\$x1 + \$x2 + \$x3". At the bottom, there is an "Input Variable References" section with an "Auto Detection" button.

Figure 10: PowerShell Objective

You may also opt to write an expression for PowerShell that references an existing objective and constraint. For example, a PowerShell objective f2 with a statement of the form ‘\$f1 – 100’ means it uses a dependency from the result of the objective f1. This feature allows you to easily do further analysis, perform transformations, or denote penalty calculations on any simulation objectives and constraints. Same as a *Babel* expressions, this will also convert referenced objective outputs as intermediate outputs.

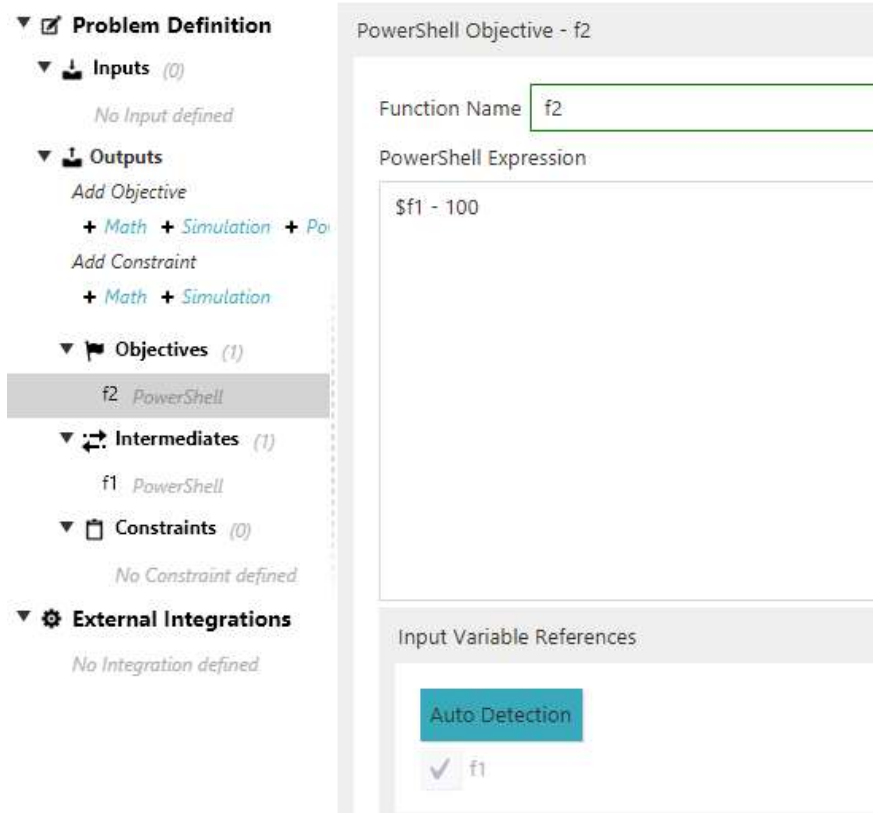


Figure 11: PowerShell Objective f2 Referencing a Simulation Objective f1

3.2.3 Simulation Outputs

Objective and constraint functions can also be the outputs of simulation models. To add a simulation objective or constraint, select + *Simulation* under *Outputs* in the *Problem Definition*. Further configuration will need to be done in *External Integrations*.

3.3 Simulation Integration

The *Simulation Integrations* section is used to define the interface between OASIS and external simulation tools. Create an integration by giving it a name, then clicking the *Add* button. Clicking on the name in the list will show detailed information about the integration. Details are defined in the following table.

Table 1: Detailed Information

Entry	Description
Command	The command-line arguments and options needed to invoke the external executable properly.
Executable Information	The location of the executable file and the length of a <i>Timeout</i> if one is chosen. A <i>Timeout</i> or <i>Max Run Time</i> in external integrations is the maximum time OASIS will spend on a single iteration before it moves on to the next.
Files	<p>The locations of the <i>Working Directory</i>, <i>Input File</i> and <i>Output File</i>.</p> <p><i>Working Directory:</i> Designates a working directory for the simulation integration. Once the working directory is specified, the paths of the <i>Input File</i> and <i>Output File</i> can be specified relative to the working directory path.</p> <p><i>Input File:</i> A file that OASIS modifies to change input parameters.</p> <p><i>Output File:</i> The output file is generated by the simulation, where objective and/or constraint function values are found.</p>

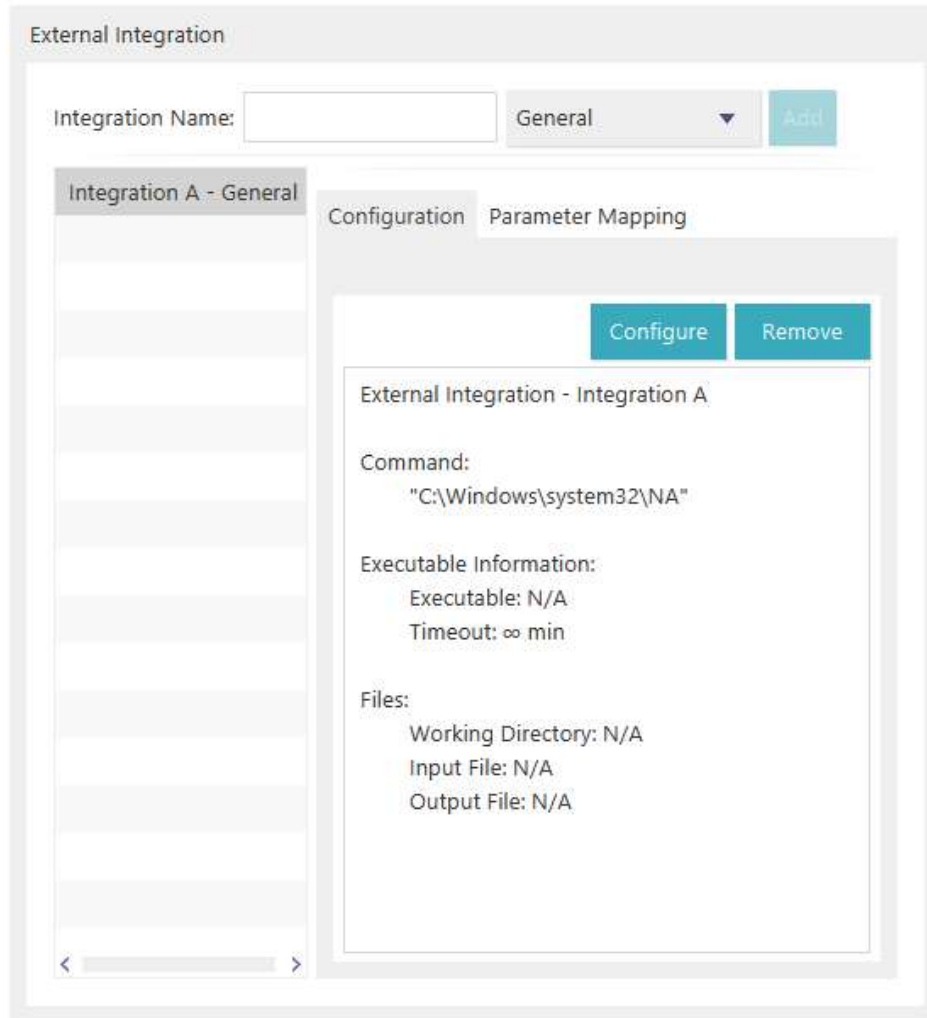


Figure 12: Integration Creation

Aside from the *Add* button, two other buttons are available: *Configure* and *Remove*. *Configure* opens a separate window in which the integration can be modified. *Remove* deletes the integration. The configurations for these external integrations can be both imported and exported via .OPYL files. The import and export options are available in the drop-down *File* menu.

3.4 Other Integrations

OASIS can be integrated with other tools via direct integration, including its API. To find out how please see empowerops.com/manual.

Simulation Integration

4 General Simulation Integration

In OASIS, using a Simulation Integration (SI) can be done either by directly integrating the SI with OASIS or by wrapping the SI using a scripting language (e.g., PowerShell). Direct integration is the preferred but, in some cases, direct integration may be too difficult, in which case a script wrapper can help overcome the integration challenges. Section 6.1 covers the usage of PowerShell as a wrapper for more flexible simulation integration.

4.1 Simulation Integration Edit Window

The SI configuration window can be accessed by clicking on the *Configure* button while an SI is selected in the Problem Setup tab. It is used to configure the interface between the SI and OASIS. This section of the manual will walk through an example configuration.

The screenshot shows the 'Simulation Configuration Window' with a sidebar on the left containing 'Executable Configuration', 'Command-line Options', and 'Parameter Capture'. The main window is divided into two sections: 'Executable Configuration' and 'Command-line Options'. The 'Executable Configuration' section includes fields for 'Executable', 'Working Directory', 'Input File', and 'Output File', each with a browse button (...). There are 'Preview' buttons for the Input File and Output File fields. The 'Backup' section has a checked checkbox for 'At Start' and an unchecked checkbox for 'Every Point', with a corresponding file path field and browse button. The 'Exit Codes' field is a dropdown menu set to 'Default (0)', with a numeric input field set to '0' and a 'Mins' dropdown. The 'Max Run Time' field is empty, with a 'Mins' dropdown. The 'Execution Strategy' dropdown is set to 'Execute by running at each iteration'. The 'Command-line Options' section contains a table with two columns: 'Options' and 'Arguments'. The table is currently empty. At the bottom of the window are 'OK' and 'Cancel' buttons.

Figure 13: Simulation Configuration Window

4.2 Executable Configuration

The *Executable Configuration* section is for defining the path to the *Executable*, paths to the *Input File* and *Output File*, and an optional *Max Run Time*. The *Input File* contains the values of the variables and any scripts or data necessary for the *Executable* to perform the simulation, and the *Output File* is generated by the simulation and contains the values of the objective and/or constraint functions. The *Max Run Time* is a setting that allows OASIS to terminate an iteration and attempt the next one if the current iteration takes much longer than expected.

1. Set the path to the executable. Either paste the path into the *Executable* text box or click the button to the right of the text box to navigate to the file. For this example, we can use the *File Watcher* simulation which is included in the *Samples* folder. A copy of the *Samples* folder must be made to somewhere there is no need for administrative computer read/write permissions, like the Desktop or a folder within Users.

Executable Configuration	
Executable	C:\Users\T\OASIS\Samples\File Watcher\File watcher script.ps1
Working Directory	C:\Users\T\OASIS\Samples\File Watcher
Input File	input.properties
Output File	
Backup	<input type="radio"/> At Start <input type="radio"/> Every Point
Exit Codes	Custom (1) 1
Max Run Time	
Execution Strategy	Execute once and monitor files for changes

Figure 14: Specification of Executable Configuration for a Simulation Integration

2. The *Working Directory* field designates a working directory for the simulation integration. Once the working directory is specified, the paths of the *Input File* and *Output File* can be specified relative to the working directory path.
3. Set the *Input File*. The input file is where the executable reads new values from. The *Input File* “input.properties” for this example is in the *Samples* folder you copied.

4. Set the *Output File*. The *Output File* is generated by the simulation and is where objective and/or constraint function values are found in. The *Output File* for the sample problem is “results.properties”.

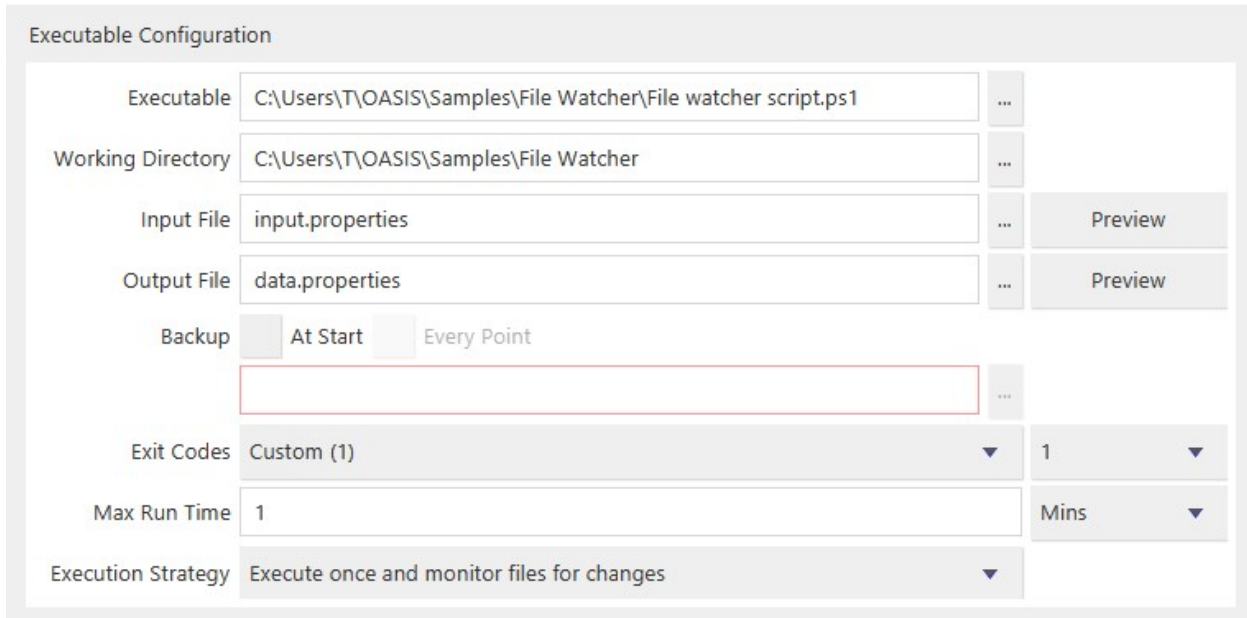


Figure 15: Specification of the Output File Path

5. You can preview the *Input File* and *Output File* by clicking on the *Preview* button next to the directory selection. A new window will pop up to allow review of the selected file



Figure 16: Preview of an Output File

6. Set the *Max Run Time* for one function call by choosing a unit of time and entering the appropriate value. This parameter is used to identify abnormal runs. Choose a conservative value (e.g. twice the expected execution time). For the sample problem, 1 minute is set for the *Max Run Time*.

4.3 Exit Codes

Exit Codes are integers that an operating system process (such as a running simulation) emits on exit. They are typically used to indicate success (by convention, exit-code 0) or failure (by convention, positive and non-zero). OASIS needs to know which exit codes represent successful runs and which represent errors or warning that can safely be ignored, so that the simulation results do not get flagged as a results from a failed evaluation. As every program has a different set of exit codes, you need to specify to OASIS which exit codes are acceptable via the *Exit Codes* section of the *Simulation Configuration* interface. The main drop-down has three choices: *Default*, *ANSYS Exit Codes*, and *Custom*.

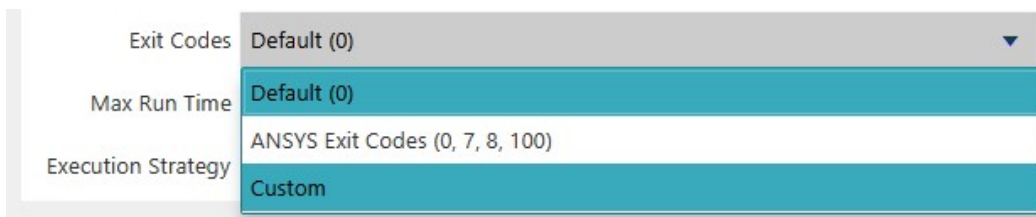


Figure 17: Exit Code Options

1. The *Default* option is appropriate for many applications where an exit code of 0 represents a successful process run.
2. Selecting the *Custom* template will enable the user to select any exit codes that they do not wish to be informed during execution. This can be done by selecting any number of exit codes from the drop-down menu to the right.



Figure 18: Custom Exit Code

4.4 Iterative Backup

When running simulation optimizations, you have the option to activate our iterative backup feature.

There are two settings for iterative backup; at the start or at every iteration. The *At Start* option is intended to backup all the files in your configured working directory when the simulation optimization begins.

The *Every Point* option may only be activated if the *At Start* option is already selected. What this feature will do is backup the files from your configured working directory at every iteration in addition to doing an initial backup at the start of the optimization.

After every iteration during a simulation optimization, the files that have been changed in your configured working directory will be restored back to their initial states except the output file. To have the output file also be restored as part of this process, you will have to use the appropriate environment variable.

-Dcom.empowerops.algorithms.ExternalToolBackupService.RestoreWorkingDirectory=value
where the value is either “true” or “false”

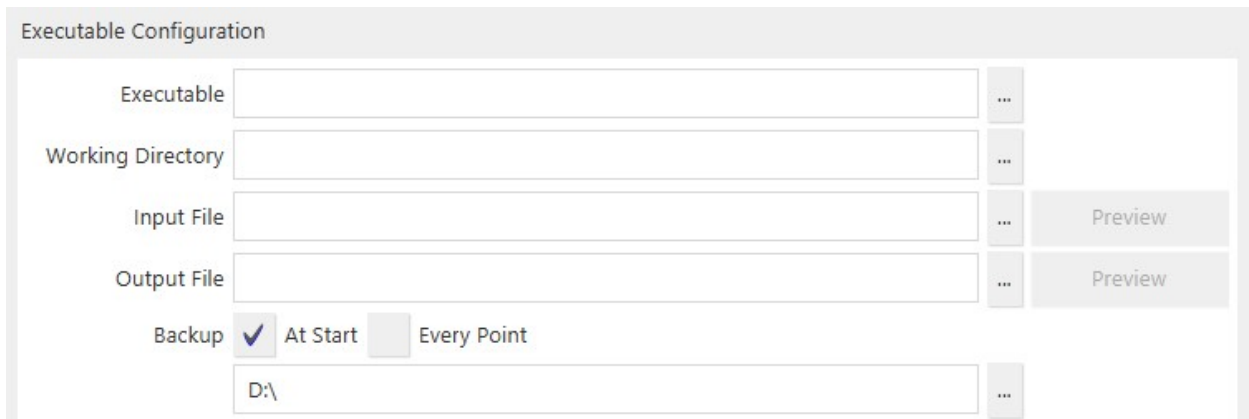


Figure 19 - Iterative Backup in Simulation Integration Settings Window

The image above shows the backup files being stored in the D:\ drive however, users may edit this path to their liking. The image below demonstrates the breakdown of files for an external simulation that is backed up at each iteration.




<input type="checkbox"/> Name	Date modified	Type	Size
 d0001	2017-05-07 3:24 PM	File folder	
 d0002	2017-05-07 3:24 PM	File folder	
 origin	2017-05-07 3:24 PM	File folder	

Figure 20 - Backup Files

Each iteration is stored in a separate folder. In the above image ‘d0001’ represents the first iteration and ‘d0002’ represents the second iteration of a simulation run. The origin folder is a backup of the files before the optimization has begun. For each iteration, relevant files are stored separately and broken up into *output*, *working dir* and *input*. The output folder stores the output file as-is generated by the simulation tool being used, *working dir* stores all the files in the working directory for said evaluation and finally, the input file is the file that was passed to the simulation tool. The figure below shows this breakup of files.




<input type="checkbox"/> Name	Date modified	Type	Size
 output	2017-05-07 3:24 PM	File folder	
 working dir	2017-05-07 3:24 PM	File folder	
 input	2017-05-07 3:24 PM	File	1 KB

Figure 21: Breakdown of Backup Files and Folders

4.5 Execution Strategy

The *Executable Strategy* has a drop-down menu for determining how OASIS interfaces with the external executable. The default option is to execute by running at each iteration, which means OASIS expects the external executable to run and exit for each evaluation. An alternative option is to execute once and monitor the file for changes (or Run-Once-Execute-Many), which means OASIS expects the external executable to run and remain open for all subsequent evaluations.

4.6 Command-Line Options

The *Command-line Options* are used to define the arguments that are required for running the executable from the command line. For this sample problem, 3 arguments are required:

Argument	Option
-i	The name of the input file. In this example, it should be the input.properties
-o	The name of the output file. In this example, it should be the data.properties
-verbose	

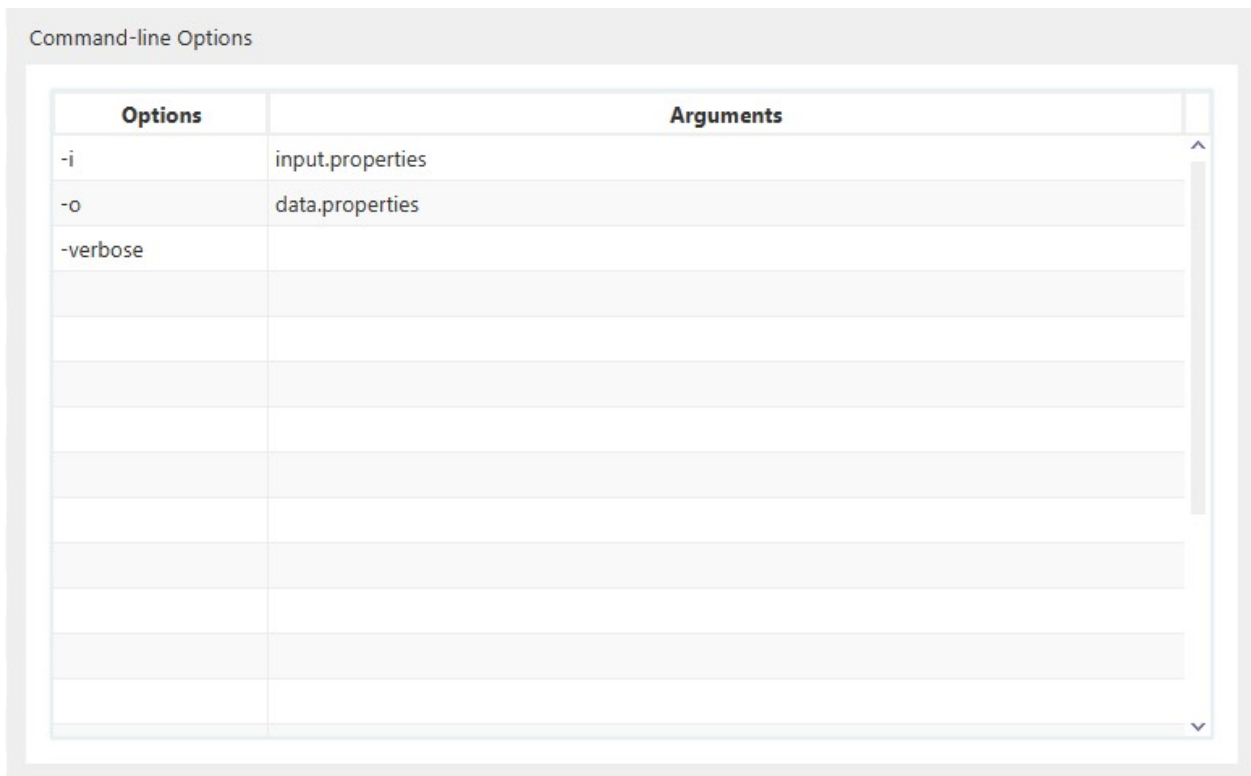


Figure 22: Command-Line Options Configuration

4.7 Simulation Parameter Capture

The *Parameter Capture* section allows you to specify the locations of the variables, objectives, and constraints in the input and output files to enable OASIS to integrate with the simulation models properly.

4.7.1 Simulation Parameter Creation

The *Parameter Capture* interface is where all external parameters are captured. On the top side input parameters are defined, and on the bottom side objectives and constraints (output parameters) are defined. To create an input or output parameter, simply press the *Add* button.

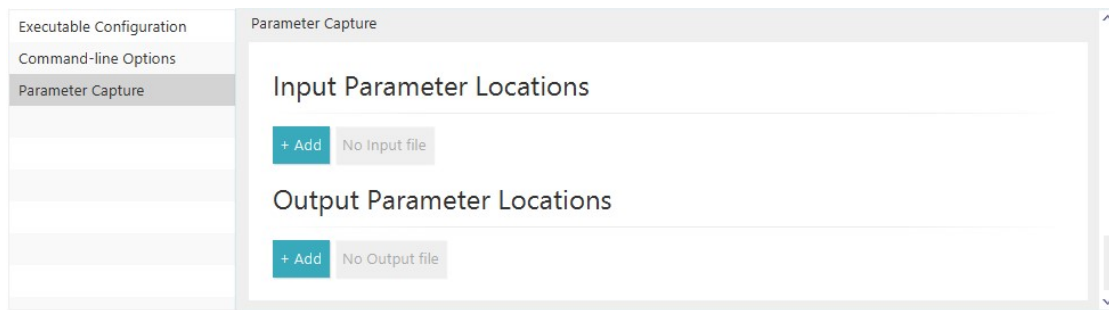


Figure 23: Parameter Capture Menu

By default, OASIS will use any existing variable, objective, and/or constraint names to name your simulation parameters. For example, if 2 variables x1 and x2 exist in the OASIS problem configuration, when a user creates a variable parameter it will be named x1, and the next one x2, and so on.

The screenshot displays two main sections: 'Input Parameter Locations' and 'Output Parameter Locations'.
The 'Input Parameter Locations' section is expanded to show a parameter named 'Height'. It includes an 'Input Name' field containing 'Height', a 'Delete' button, and an 'Add location' dropdown menu with the text 'Select locator step type...'. Below this section are '+ Add' and 'No Input file' buttons.
The 'Output Parameter Locations' section is expanded to show a parameter named 'Result'. It includes an 'Output Name' field containing 'Result', a 'Delete' button, and an 'Add step' dropdown menu with the text 'Select locator step type...'. Below this section are '+ Add' and 'No Output file' buttons.

Figure 24: Adding Parameters

4.7.2 Simulation Parameter Capture Steps

Naming your parameters is not enough to link OASIS to your simulations. The value of each variable needs to be written to specific locations in the *Input File* at every iteration, and the objective and constraint values need to be read from the *Output File*.

The locations of input parameters in the *Input File* are specified by designating a range of characters to replace with the new values that OASIS will generate via a set of indices. You can quickly do this using the *Detection Window*. The locations of output parameters in the *Output File* are specified by giving OASIS an *Anchor Text* to look for. OASIS will grab the numbers following the *Anchor Text* once it is found.

If an input variable appears more than once in the *Input File* and all instances of it need to be replaced per iteration, multiple *Specific Text Location* tabs can be added per input variable the same way the first one was created.

Output variable

The option to search a file based on specific landmarks in it are referred here as *Anchors*. Multiple *Anchors* can be read through in sequence to reach very specific areas of a file. If there are multiple instances of an *Anchor* in the file, the *Occurrence* field on the *Anchor* will allow the user to specify which instance they would like to read from.

Some subtleties to remember when creating *Capture Steps*:

- Each additional Objective or Constraint Capture Step will start searching the file starting at the end of the last Step.
- Indices in the Parameter Detection steps are incremented from one, so the first row in the file will be row '1', and the first character in a row will be in column 1.
- Objective and constraint steps which do not find a usable value will be treated as an inconsistent point and be given an infinite value.
- Anchors are case-sensitive.

4.7.3 Detection Window

The way to create external parameters is to use the *Detection Window*. This interface can be accessed via a button on the bottom of the *Parameter Capture* section in the simulation integration window. There is a button to access the window for the *Inputs*, and one for the *Outputs*.

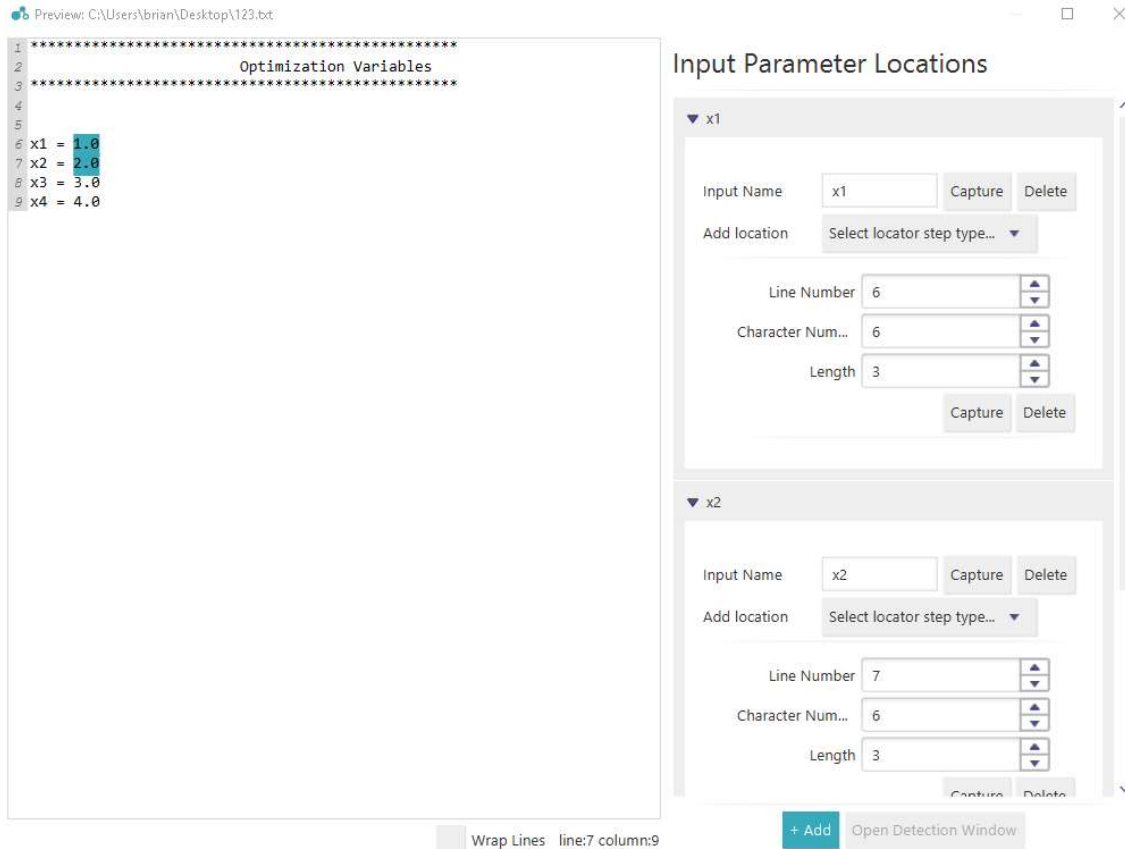


Figure 25: Detection Window for Input File

This UI allows you to *Capture* various values directly from the *Input File*. Capturing is the process of selecting an area of the preview and having the selected location information be automatically saved into the parameter capture text fields.

Let's go through a quick example to capture the values for the x5 variable again.

1. Begin by creating a new Variable with the + *Add* button.
2. Next, we need to name our new Variable, and while we could simply type "x5" in manually, this gets annoying for longer variable names. Instead, click the *Capture* button next to the *Input Name* field that you want to edit, and then highlight the name of the variable if it appears in the *Input File*.
3. Finally, we need to add a *Specific Text Location* which tells OASIS where to write this variable's values to within the *Input File*. Click *Capture* and then highlight the numbers in the desired location. The *Line Number*, *Character Number*, and *Length* boxes will populate themselves automatically afterwards.

This process is similar in the *Output/Constraint Detection Window*. Anchors are used to reliably traverse *Output Files* when they change drastically from one iteration to another. The following example captures the outputs:

1. To *Capture* the value of *Result*, we must first create an output parameter locator by clicking on the + *Add* button towards the bottom of the *Detection Window*.

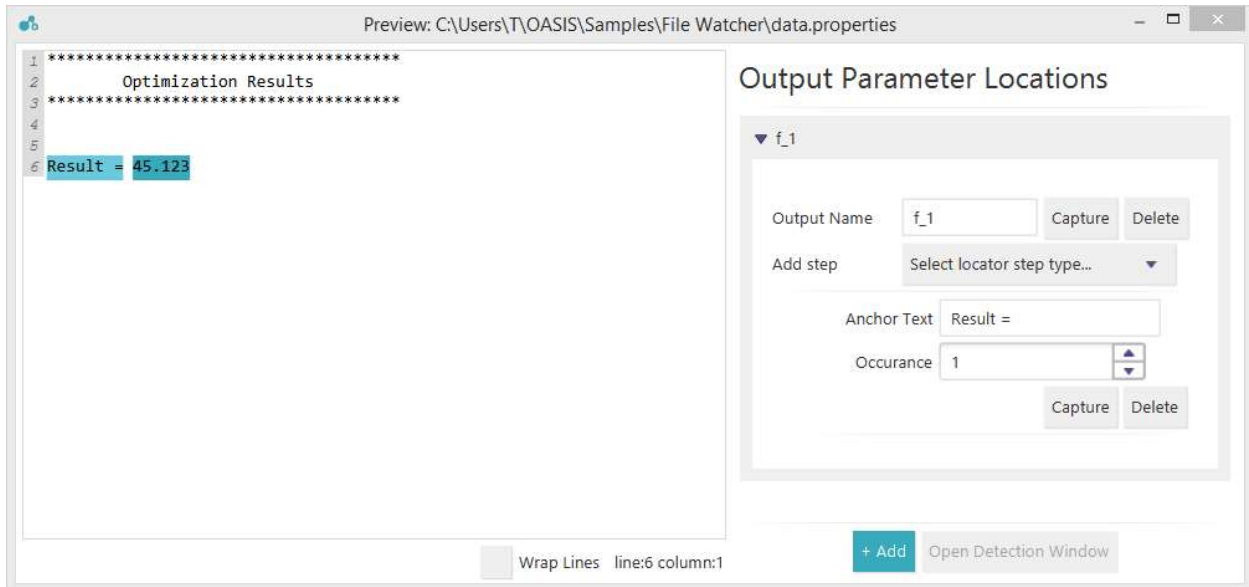


Figure 26: Preview Window for Output File

2. Type the *Output Name* or *Capture* it from the file.
3. Then *Add step* and pick *Look for known `Anchor` Text*. Click *Capture* and highlight the desired *Anchor* text. In the figure above, it was “Result =”. The numbers following the *Anchor* text are recognized as that output’s value.

4.8 Simulation Integration MATLAB Example

In this example, OASIS will interface with MATLAB, with a 10-variable function implemented in MATLAB serving as the simulation integration. All files needed for this example can be found under \OASIS\Samples\MATLAB folder in the Program Files folder. If your machine is x64, then the OASIS folder is most likely in the Program Files (x86) folder. To run the MATLAB example, perform the following steps:

1. Copy the MATLAB sample problem folder to another directory, for example your Documents folder. All references in the subsequent steps refer to the new, copied MATLAB sample problem folder.
2. Launch OASIS.
3. Navigate to the File menu, *Import from OPYL file...*, and select the *MATLAB example.opyl* file in the MATLAB sample directory. Select *Import*.
4. In the main OASIS window, open the Simulation Integration and double click on the MATLAB tool to begin editing.
5. Set the *Executable* path as the MATLAB.exe file. It should be in the “bin” folder in the MATLAB directory in Program Files.
6. Set the *Input File* path as the “yourscript.m” file in the MATLAB sample problem folder.
7. Set the *Output File* path as the “data.txt” file in the MATLAB sample problem folder. This file may not exist, but the path should still be specified.
8. In the *Command-line Options*, in the *Argument* for the -r option, replace ‘[replace with MATLAB Sample directory in the OASIS Samples folder]’ with the working directory.
9. Close the Simulation Configuration window and click *Optimize* in the OASIS window.

4.9 Command Line Options Configuration

The Command Line Options are used to define the arguments that are required for running the executable from the command line. When using the OASIS interface with MATLAB, there are a few arguments recommended. These arguments are listed as: “-nosplash”, “-wait”, “-automation” and “-r”. Firstly, the “-nosplash” argument is used to disable the splash screen MATLAB displays when the application is started. Next, the “-wait” and “-automation” arguments are to allow MATLAB to automate and run each function evaluation, and have OASIS wait while this process is taking place. Finally, the “-r” argument is used to declare the directory where each MATLAB “.m” files exist in and is also used to run the MATLAB executable and have it exit after each evaluation. An example parameter for the “-r” argument would be as follows: “cd('directory'); yourscrip; exit”.

4.10 File Watcher SI Execution Strategy

The default SI execution strategy is to “execute by running at each iteration” where the SI process is invoked for every candidate solution. This continual start up and tear down of the SI process is expensive and avoiding it will save execution time. The File Watcher Execution Strategy (also known as Run-Once-Execute-Many) is an alternative and more efficient way of executing simulation integrations where the SI process is invoked once and remains running to execute multiple candidate solutions.

This option configures OASIS to run your simulation integration once and wait for file changes, which means OASIS expects the simulation integration to run at the beginning of the optimization and remain open for all subsequent candidate point evaluations. With this option, OASIS will write variable values to the input file specified in the simulation integration configuration window, start the SI process, and waits for changes to be made to the output file. Meanwhile, the SI process should detect a change in the input file, which will trigger it to read the input file, run its analysis, and then write the result to the output file. OASIS sees the change to the output file and reads the output file to obtain the results. See Figure 27 for an illustration of this process.

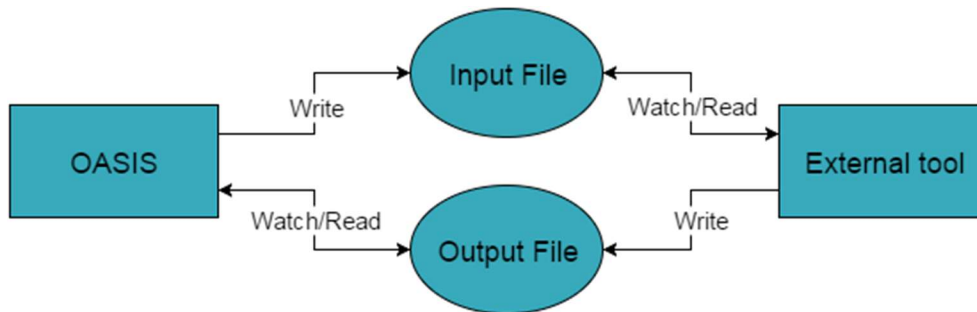


Figure 27: File Watcher Process

4.10.1 Configuring Your Simulation Integration to Run with File Watcher Strategy

For the File Watcher strategy to work, your SI needs to be configured to operate according to the process outlined in the previous section. Your tool must be configured to watch a file, read the file, run the analysis, then write the results to file. This configuration can be achieved via a script, and the following are the recommended steps that the script should follow:

1. Watch the input file whose path was provided to you.

2. When a change is detected, read the input file.
3. Start the analysis with the data from the input file.
4. Write the results of the analysis to the output file whose path was provided to you.
5. Return to step 1.

For an example of what a script following the above steps looks like, see the file “filewatchertestscript.ps1” in the following location:

Program Files (x86)\OASIS\Samples\File Watcher

4.10.2 File Watcher Example

In the following example, OASIS will interface with PowerShell using the File Watcher Execution Strategy, with a small function written in PowerShell to function as the Simulation Integration. All files needed for this example can be found in the following directory:

Program Files (x86)\OASIS\Samples\File Watcher. To run the example, perform the following steps:

1. Copy the *File Watcher* sample problem folder to another directory, for example your Documents folder. All references in the subsequent steps refer to the new, copied sample problem folder.
2. Launch OASIS.
3. In the File menu, click on *Import from OPYL file...* and import the *File Watcher.opyl* file in the File Watcher folder.
4. Deselect the *Simulation Integration*, this example involves creating it manually.
5. Click on *Simulation Integration* and add a *General* integration called File Watcher.
6. Click on *File Watcher* and then click the *Configure* button.
7. Set the *Executable* path as the “File watcher script.ps1” file in the File Watcher folder.
8. Set the *Working Directory* to the File Watcher folder.
9. Set the *Input File* path to be the *input.properties* file in the File Watcher folder.
10. Set the *Output File* path to be the *data.properties* file in the File Watcher folder. This file may not exist, but the path should still be specified.
11. Set the *Execution Strategy* to *Execute once and monitor files for changes*.
12. Add a *Command-line Option* named *-i* with an argument that is the same as your input file name (input.properties).

13. Add another *Command-line Option* named `-o` with an argument that is the same as your output file name (`data.properties`).
14. Scroll down to the *Parameter Capture* section.
15. Under the *Input Parameter Locations* heading, click the *Open Detection Window* button.
16. Click *+ Add*.
17. In the `x_1` panel that appears, click the dropdown menu beside *Add location* and select *Specific text location*.
18. Click *Capture* and highlight the value that comes after “`x_1=`” in the textbox on the left side of the screen.
19. Repeat Steps 16 to 18 for `x_2` and `x_3` but highlighting the appropriate values in Step 18 (i.e. the value that comes after “`x_2=`” for `x_2` and the value that comes after “`x_3=`” for `x_3`).
20. Close the *Preview* window.
21. Under the *Output Parameter Locations* heading click the *Open Detection Window* button.
22. Click *+ Add*.
23. In the `f_1` panel that appears, click the dropdown menu beside *Add step* and select *Look for known ‘Anchor’ text*.
24. Click *Capture* and select “`f_1=`”, then close the *Preview* window.
25. Click OK. The *Optimize* button should not be reporting any issues.
26. Click *Optimize*.

The *Simulation Integration* configurations done in the steps above are already included within the *File Watcher.opyl* file. If Step 4 above is not followed and the *Simulation Integration* is imported, the sample will be ready to run right away.

5 Multi-Simulation Configuration

In OASIS, a user may also set up problems that contain chained simulations, or multiple simulation integrations. To clarify, these types of problems require more than one simulation tool to solve a problem usually with subsequent simulation tools depending on the results from previous simulations tools (i.e. the results from simulation integration #1 are used as inputs to solve for the results for simulation integration #2). The following example will illustrate how the user may look to configure this type of problem in OASIS:

1. The first step will be for the user to add all the necessary variables. We recommend users to employ a naming convention that will help identify variables to their respective simulation tools. Doing this will ensure mapping of the parameters in the upcoming steps are not as difficult. If possible, please use similar naming conventions when creating objectives and constraints as well.

Name	Min Val	Max Val	Step Size
et1_var1	11	20	0
et1_var2	11	20	0
et2_var1	0	1	0
et2_var2	0	1	0

Figure 28: Variables Named to Match Simulations

▼ **Outputs (2)**
 Add Objective
 + Math + Simulation + PowerShell
 Add Constraint
 + Math + Simulation

▼ **Objectives (2)**
 et1_obj Simulation
 et2_obj Simulation

▼ **Intermediates (0)**

Figure 29: Objectives Named to Match Simulations

2. After the relevant variables, objectives and constraints are set up the user may begin to create each simulation tool as necessary.
 - Note that when mapping parameters to match only the variables used in a simulation to their counterparts. The user will also see objectives listed as an input. This is intentional because in certain problems the objectives of a simulation are used as an input in another simulation.
 - Mapped parameters will show up green and will not be allowed to be matched.

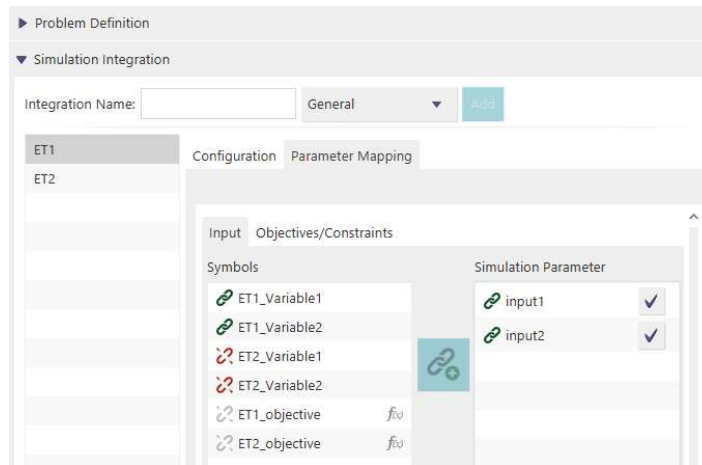


Figure 30: Parameters Mapped Correctly

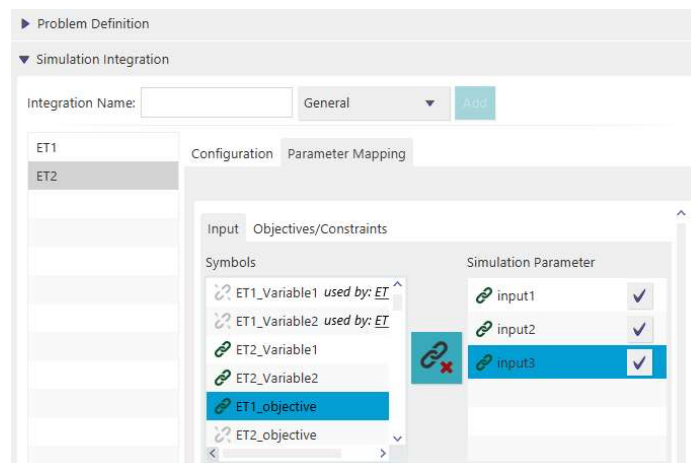


Figure 31: Input 3 for Second Simulation Matched with Output of First Simulation

6 PowerShell Integration

6.1 PowerShell Simulation Integrations

PowerShell is a powerful scripting language that can be used as a wrapper around the SI to make the integration process with OASIS easier. PowerShell is a free add-on to any version of Windows from XP and onwards. It enables Windows users to take advantage of a flexible prompt with an extensive scripting support. With Windows PowerShell, a script can be written that will perform nearly any task windows itself is capable of, such as transforming data between file formats or executing a tool that may not be runnable directly on the command line.

6.1.1 PowerShell Setup

If PowerShell 4.0 is not available on your machine, go to the following link and follow the instructions to [install PowerShell](#):

Microsoft also has a [PowerShell owner's manual](#) that describes running a PowerShell script for the first time.

6.1.2 Creating Custom PowerShell Scripts

Custom PowerShell can be very useful for running more complex tools that require more maintenance and attention than can be configured purely through command-line options. There are a few things to remember when creating custom scripts that are executed at each iteration.

Inputs can be Accessed Directly or Read from the Input File

OASIS passes the variables via the command line when invoking the PowerShell script. The user can write a 'Param()' declaration block and directly reference any of the simulation integration variables from within their PowerShell script. For example, if SI variables 'x1' and 'x2' were created for use in a PowerShell file, you could access their values in the PowerShell script with '\$x1' and '\$x2' by making them available with the following declaration at the top of your script:

```
Param(  
    [Parameter(Mandatory=$true)][Double] $x1,  
    [Parameter(Mandatory=$true)][Double] $x2  
)
```

The input file mechanism for simulation integrations also works for PowerShell scripts, and the input file will be re-generated for each iteration. To parse input values from an input file named “inputs.txt” containing entry lines such as ‘x1=1.234’, ‘x2=5.678’, the following commands can be used:

```
$inputFile = "inputs.txt"  
$fileContent = Get-Content $inputFile -raw  
$inputs = ConvertFrom-StringData $fileContent  
  
[Double] $x1 = $inputs["x1"]  
[Double] $x2 = $inputs["x2"]
```

Outputs Should Be Written to a File

To write the generated output, for example ‘f1’, to the output file, the following commands can be used:

```
"f1=" + $f1 + "`n" | Out-File -Encoding Ascii "results.txt"
```

This will write the output in the format of ‘f1=2.15462’ to ‘results.txt’. Note that ‘`n’ (backtick - n) represents a new line character in PowerShell.

Example

The code below is a complete simple PowerShell script which executes a mathematical function. ‘\$x1’ and ‘\$x2’ are made available with the ‘Param()’ declaration. The result of this function is written to the output file.

```
Param(  
    [Double] $x1 = $(throw "x1 value was not provided"),  
    [Double] $x2 = $(throw "x2 value was not provided")  
)  
  
$sc_ps_result = 4*$x1*$x1 - 21/10*[System.math]::pow($x1, 4) +  
1/3*[System.math]::pow($x1, 6) + $x1*$x2 - 4*[System.math]::pow($x2, 2) +  
4*[System.math]::pow($x2, 4)  
  
"sc_ps_result=" + $sc_ps_result + "`n" | Out-File -Encoding Ascii "results.txt"
```

6.1.3 Simulation Integration PowerShell Example

To use PowerShell with OASIS, write the PowerShell script so that it uses the same variables as defined in the optimization problem and to write the output results to the output file specified in the *Simulation Integration* configuration UI. Use the path to this PowerShell script file as the executable path as well as the path to the input file.

In this example, OASIS will interface with PowerShell, with a small function written in PowerShell as the Simulation Integration. All files needed for this example can be found under ‘OASIS\Samples\PowerShell\SC function’ folder in the Program Files folder. If your machine is x64, then the OASIS folder is most likely in the Program Files (x86) folder. To run the example, perform the following steps:

1. Copy the 'SC function' sample problem folder to another directory, for example your Documents folder. All references in the subsequent steps refer to the new, copied sample problem folder.
2. Launch OASIS.
3. In the File menu, click on “Import from OPYL file...” and import the “SC example opt config.opyl” file in the SC function folder.
4. Deselect the *Simulation Integration*, this example involves creating it manually.
5. Click on the *Simulation Integration*.
6. Add a General integration called SC Function.
7. Click on your new integration and click the *Configure* button.
8. The Simulation Integration section will be a little bit different than most other simulation integrations in PowerShell's case. In most other Simulation Integrations, the executable would be an executable program, but in this case, the executable will be the ‘sc-like polynomial.ps1’ script file. No Command Options are necessary.
9. Single-run PowerShell does not require an *Input File* because the parameter information is passed directly to the script.
10. The *Output File* should be set to the *results.txt* file in the SC function sample problem folder. This file may not exist, and in this case will be created when the optimization starts.
11. Press *OK* at the bottom of the *Simulation Integration Configuration* window.
12. The *Optimize* button should not be reporting any issues.
13. *Optimize*.

6.1.4 Using PowerShell with a Remote Machine or an HPC Cluster

The user can also write a PowerShell script to perform computations on a remote machine. In the script, the user can transmit the input files to the remote machine and invoke computation commands over a Secure Shell (SSH) session. The outputs can then be read from the Standard Output (STDOUT) of the remote session or retrieved in a file. The example in ‘OASIS\Samples\PowerShell\SC function - Putty’ demonstrates the evaluation of an expression remotely through PuTTY with SSH.

The user may wish to perform resource-intensive and time-consuming computations on a more powerful remote HPC cluster. The example in ‘OASIS\Samples\PowerShell\HPC example’ comes with sample scripts that can be used with an HPC environment using the TORQUE Resource Manager/Job Scheduler. The PowerShell script ‘client_side.ps1’ first sends the input file with PSCP (PuTTY’s implementation of Secure Copy) to the remote machine, which is typically a login server, and then invokes ‘job_runner.sh’ to submit a job, which in turns calls ‘file_monitor.sh’ to track the status of the job by monitoring the output file. ‘test.sh’ contains the job content to be evaluated by one or more worker nodes of the HPC cluster.

7 Symbol Mapping

In OASIS, there are two classes of *symbols*: optimization symbols and simulation parameters. Optimization symbols are the *Inputs*, *Outputs*, and *Constraints* defined via the *Problem Definition* pane. Simulation parameters are the *Inputs*, *Outputs*, and *Constraints* defined from the *Simulation Integration Configuration*. These two classes of symbols are defined separately and need to be mapped together prior to running optimizations with external simulations. By default, all symbols created are turned on. You may opt to turn off symbols that have been created but are not used in an optimization. Symbols that are turned on will be coloured and symbols that are turned off will be greyed out. Turning on or off symbols may be done via the check-boxes on the right side of the *Simulation Parameter* table.

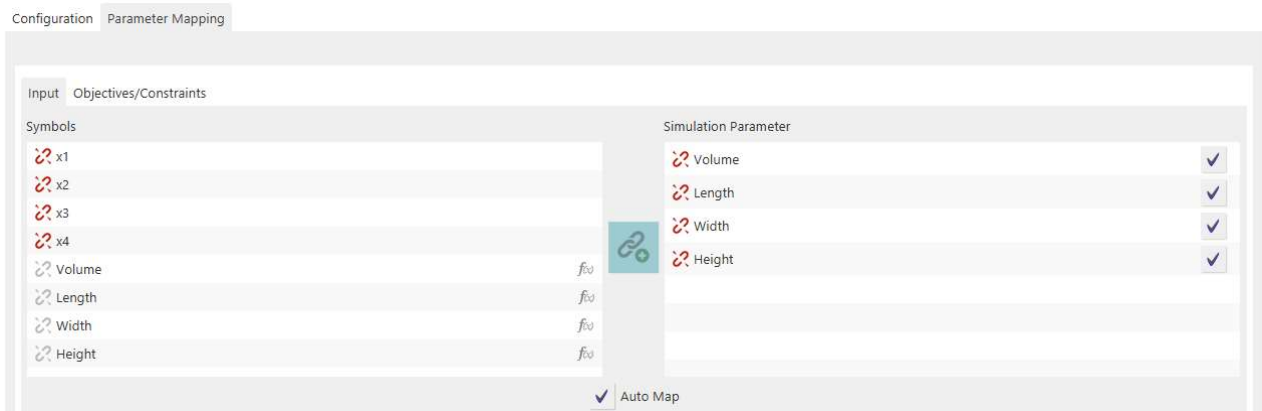


Figure 32: Symbol Mapping with All Simulation Parameters Turned On

7.1 Symbol Mapping States

Symbol mapping states helps you identify optimization symbols that are mapped to more than one simulation tool. This feature is relevant for when you are setting up a complex problem that involves more than one simulation tool, and for variables, objectives, and constraints that are used across these tools. There are two general states you should look out for: *used by*, and *also used by*. A status is displayed beside all linked optimization symbols which indicates the simulation tool it is mapped in.

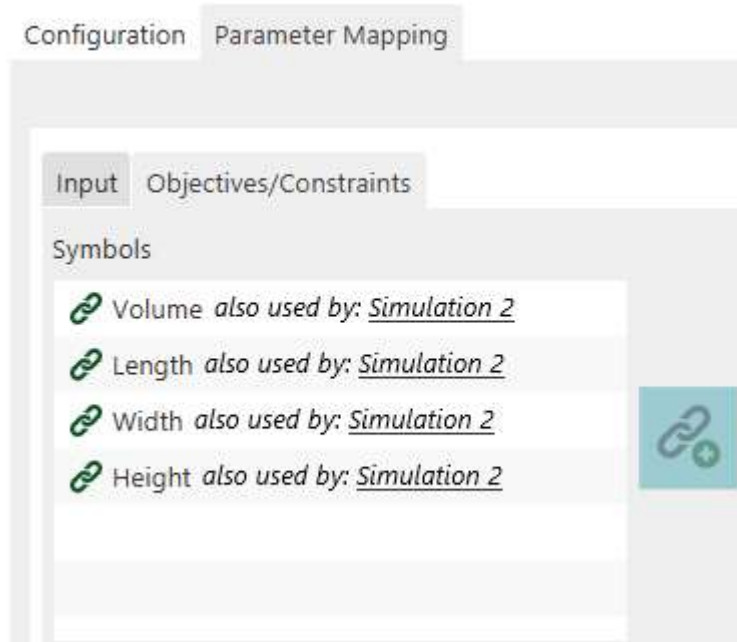


Figure 33: Optimization Symbols Mapped in One Simulation

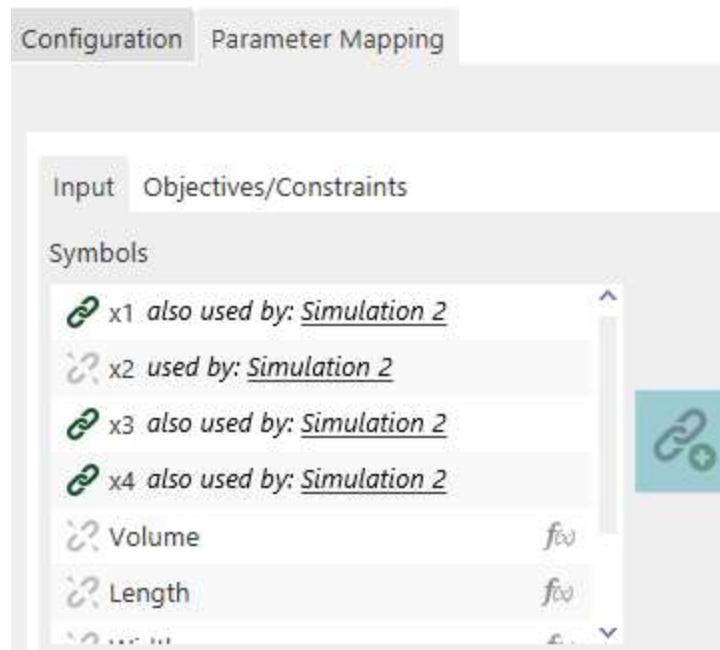





Figure 34: Optimization Symbol Mapped in More Than One Simulation

7.2 Create Mapping

If the *Auto Map* feature is active, variable symbols will be automatically mapped to simulation parameters when there are matching names. Mentioned in Section 4.7.1, when new simulation parameters are created they will automatically be named based on the next available variable,

objective, and/or constraint names. The *Auto Map* feature may be deactivated by de-activating the check box. When the *Auto Map* feature is disabled, a user may manually create links between two symbols by selecting one symbol from each table and then then clicking the *Link*  button in the middle of the interface. Once linked, a new entry will be added to the *Mapping Table* and each symbol's status will show the *Green Linked Icon* . You can only map between symbols that are not already mapped. Unmapped symbols are marked with the *Red Unlinked Icon* .

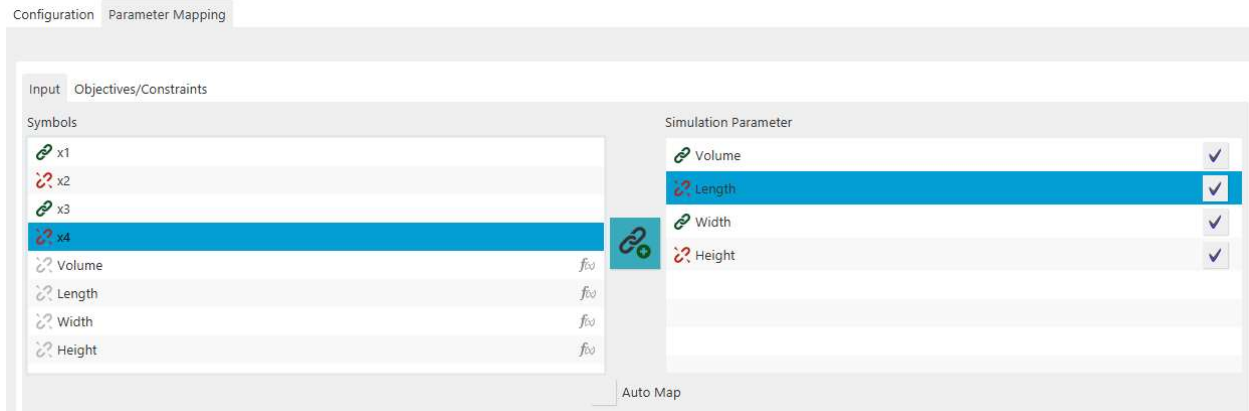


Figure 35: Part I, Icon and button state when available symbol can map

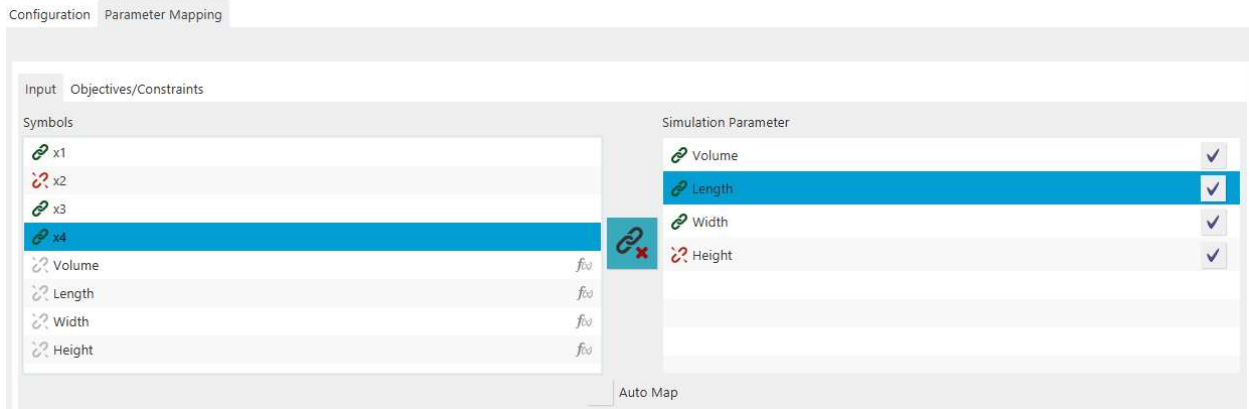
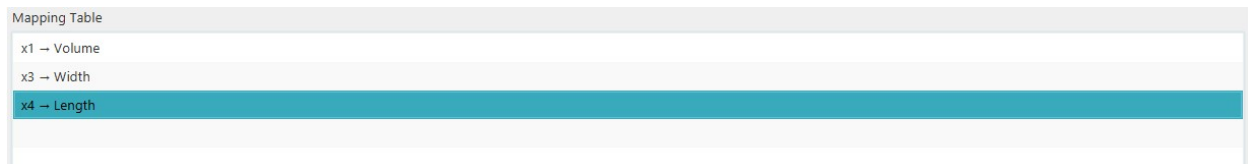



Figure 36: Part II, State when symbol is mapped



Mapping Table	
x1	→ Volume
x3	→ Width
x4	→ Length

Figure 37: Part III, Mapping Table with the New Entry Added

7.3 Remove Mapping

You can remove the established link between two symbols and make them available to link to other symbols. First, click on either of the symbols you want to unlink (alternatively, you can click on the relevant entry in the *Mapping Table*). The link will be highlighted, and you will be able to click on the *Break Link Button*  to break the link.

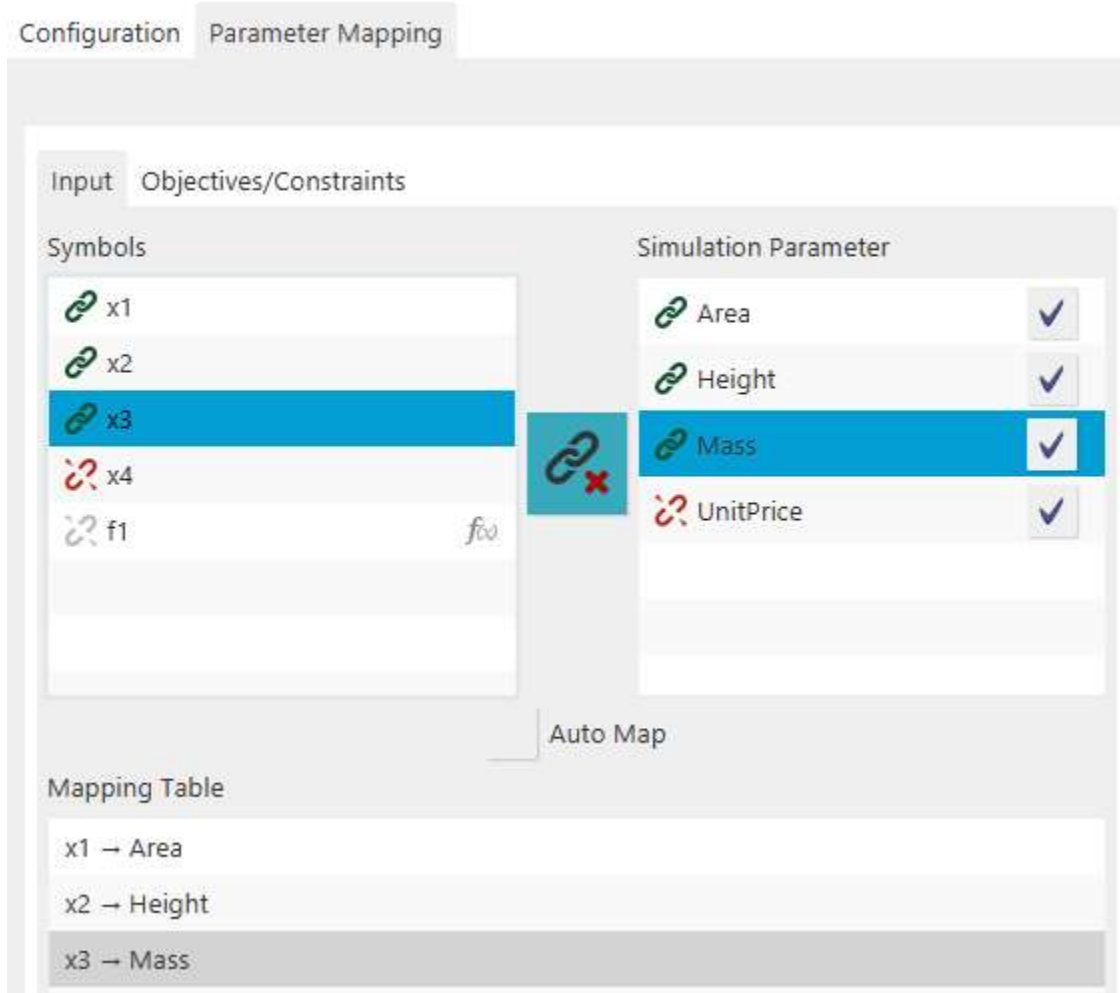


Figure 38: Icon and Button State when Selected Map Entry can be Unmapped

Optimization and Parametric Analyses

8 Design Verification

8.1 Verify Expression

OASIS allows Babel expressions to be tested at different input values. Input values used for evaluation can be added in *Design Verification* within the *Module Setup* tab.

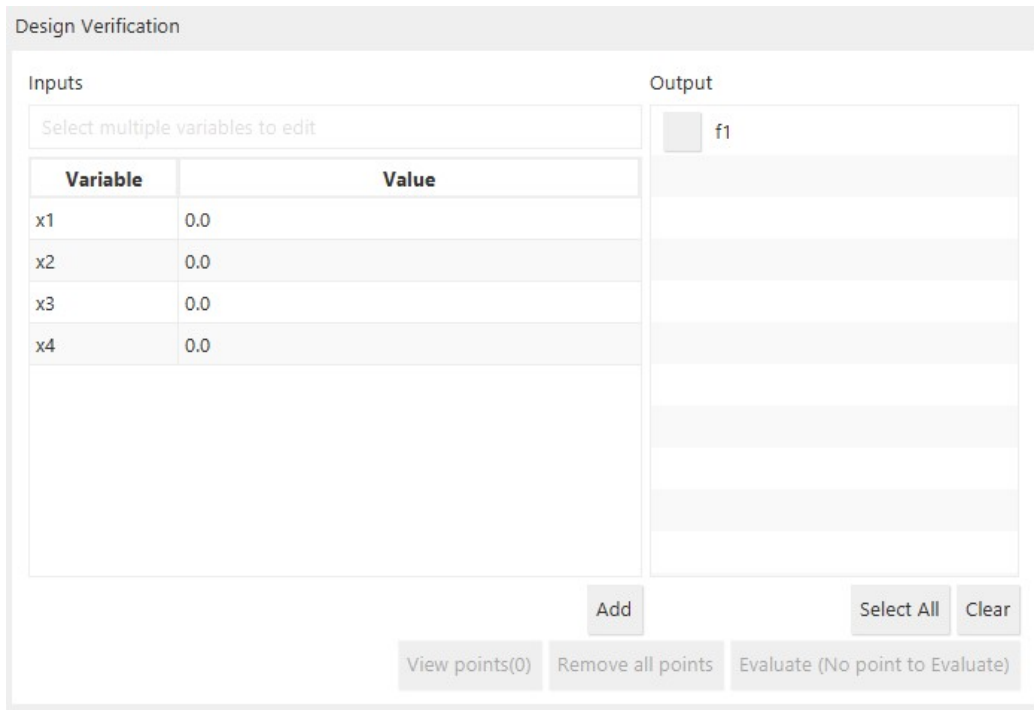


Figure 39: Design Verification in OASIS

To add an input value to a *Variable*, simply click on its respective cell in the *Value* column and type the number. After, press *Enter* on the keyboard or click on another variable's *Value* cell.

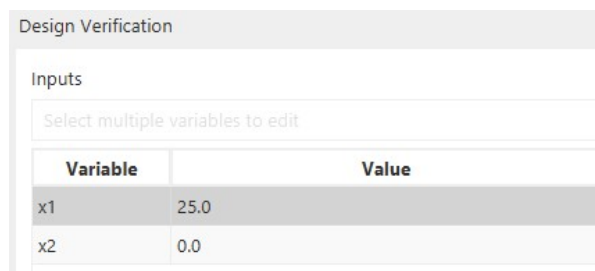


Figure 40: Adjust Input Values

To modify multiple input values, select the desired rows using CTRL/SHIFT selection techniques and then enter the desired value in the cell directly underneath *Inputs*.

When there is a problem with the expression, an error message will be displayed (see the table below).

Table 2: Error Messages for Babel Verification

Message	Description
Expression is invalid	Expression is not legal babel expression and cannot be evaluated
There is an error occurred during evaluation	Expression cannot be evaluated. This happens during dynamic variable access, usually when asking for an out-of-bound index

8.2 Simulation Verification

Before running an optimization with external simulations, we advise users to consider using the simulation verification tool to verify the correctness of the problem setup. The simulation verification tool is labeled *Design Verification* in the *Module Setup* tab of OASIS. The simulation verification tool allows users to run the simulation with one or more desired design points. The flow of this is separate from starting an optimization run because the user controls the design points used. In addition, unlike an actual optimization run, OASIS stores metadata associated with the simulation run in the working directory which enables easier diagnosis of a failure.

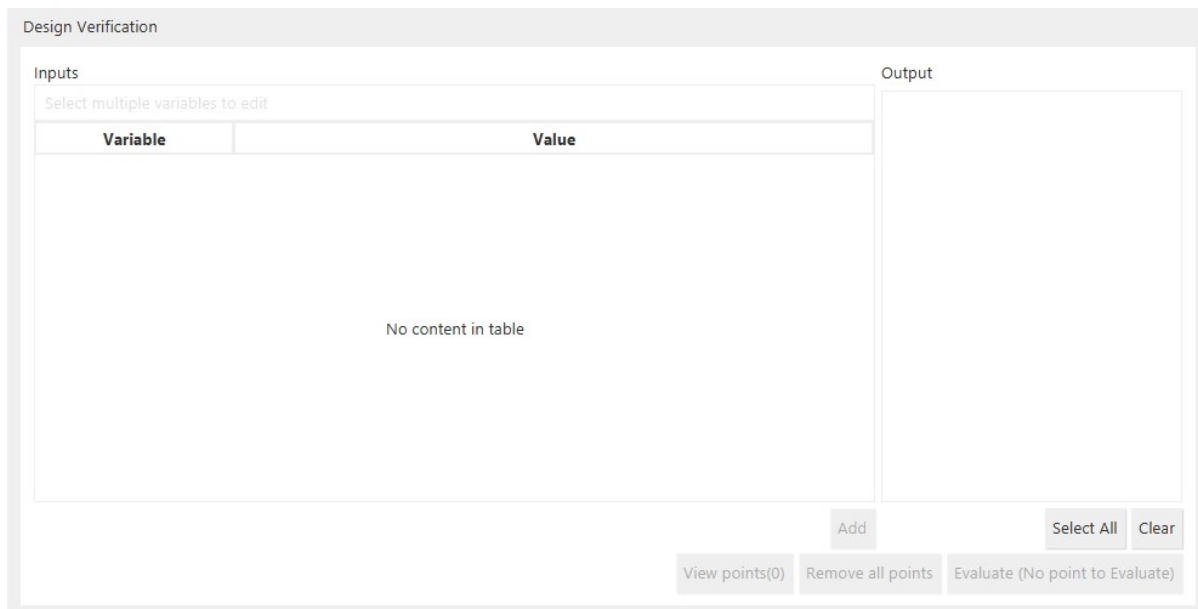


Figure 41: Simulation (Design) Verification

After the problem definition is set up the user will see all of the relevant variables under the *Inputs* section and all of the relevant constraints and/or objectives in the *Output* section. The user may then add in any desired values for the variables. Click the *Add* button to save the design points into the simulation verification tool.

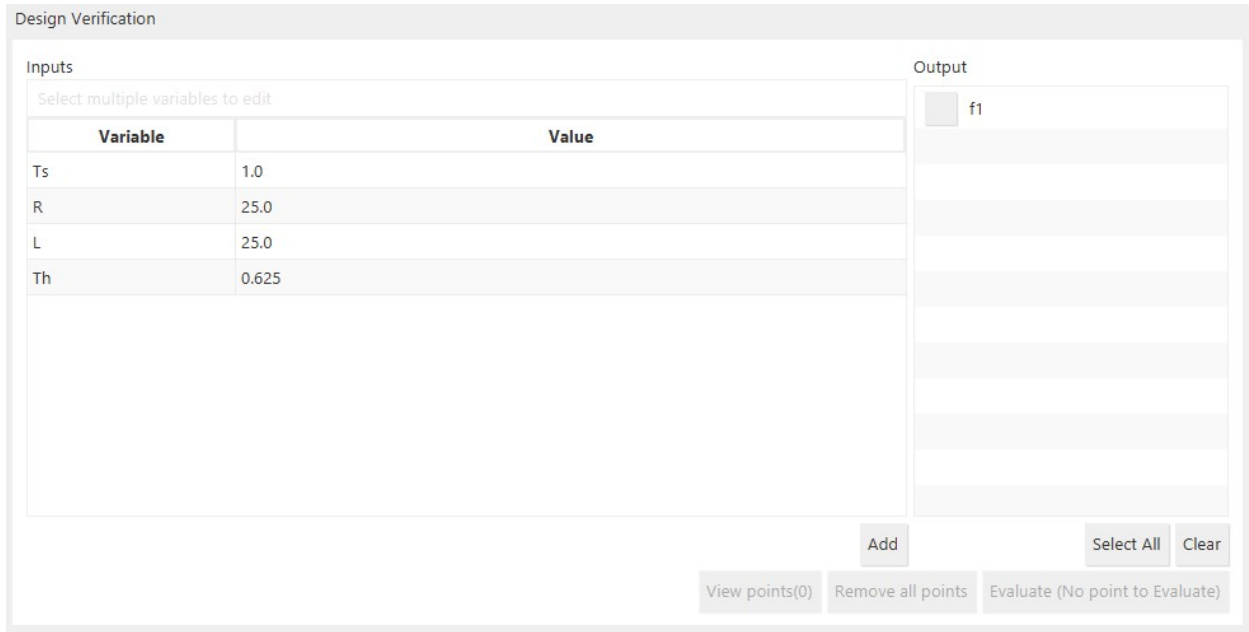


Figure 42: Design Points for a Problem

To view saved design points, click *View points*. If there are no points saved, the button will be un-clickable. The user may also remove all the saved points by clicking the *Remove all points* button.

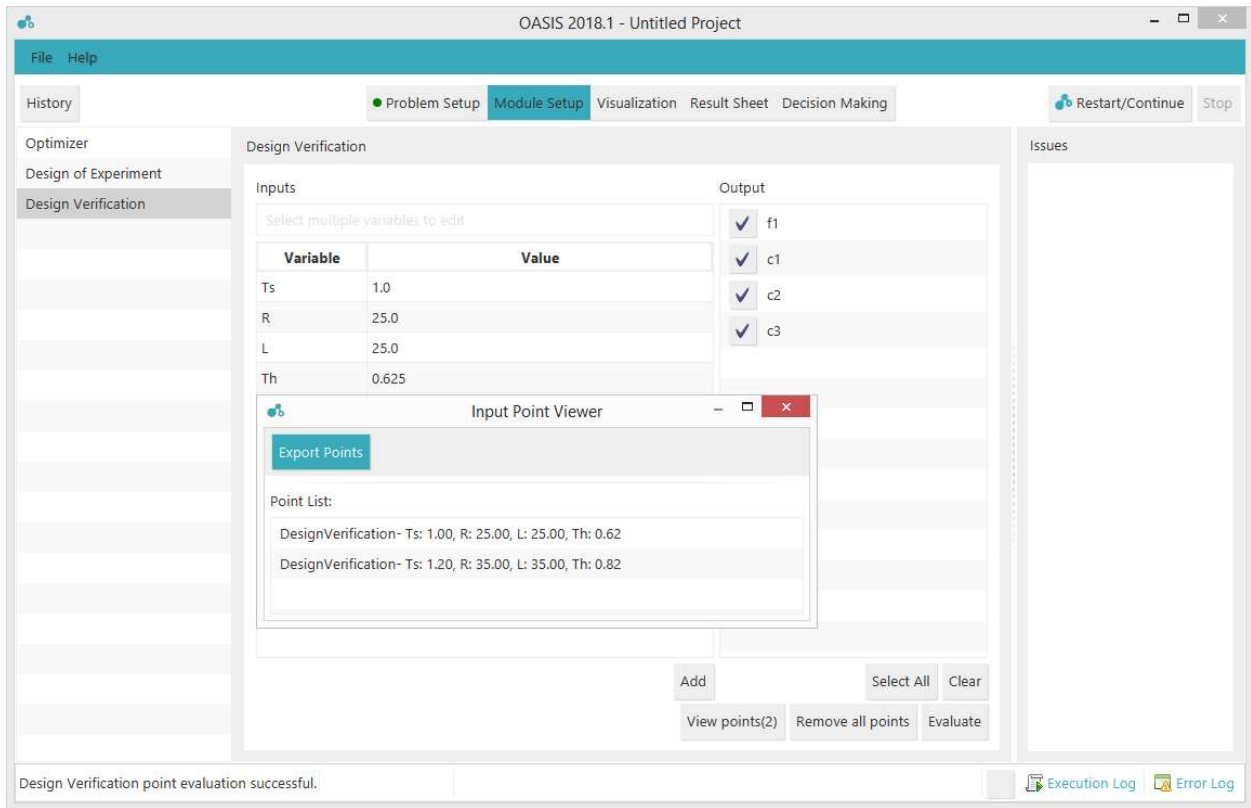


Figure 43: View Saved Design Points

Outputs will need to be specified before the user can click on the *Evaluate* button. The above example uses one output therefore only *f1* may be selected. For a problem containing multiple objectives the user will be allowed to mix and match between all the available output options. Once the simulation run has completed the results will be displayed in the same format as a regular optimization run. The user will be brought to the visualizations area and the *Results Sheet* will be available as well.

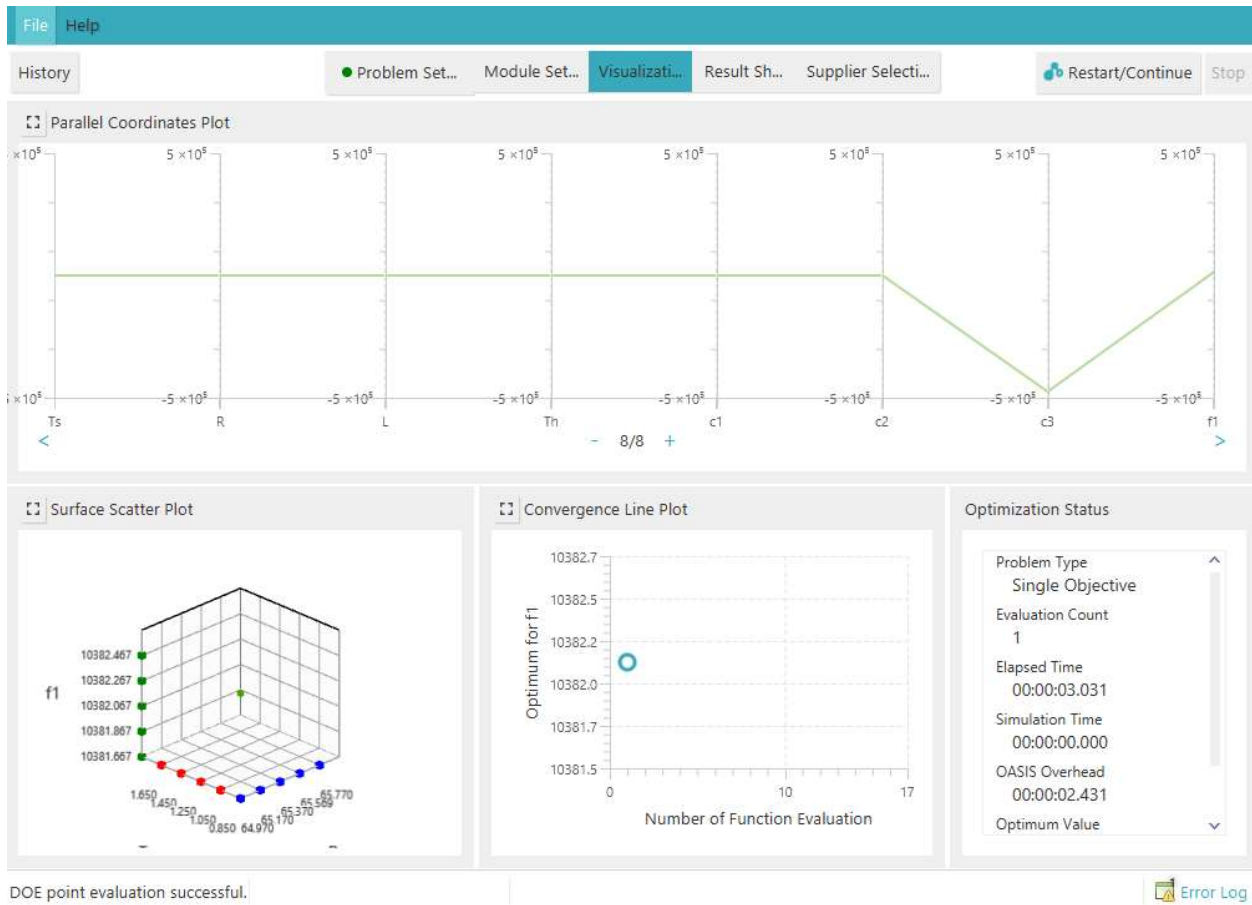


Figure 44: Simulation Verification Run Results

9 Optimization

9.1 Optimizer

The *Optimizer* tab allows configuration of optimization convergence and run settings. Details of each setting are explained below.

The screenshot shows a settings panel titled 'Optimizer'. It contains four rows of settings, each with a grey square icon to its left and a corresponding input field to its right:

- Number of runs:** Input field contains '1'.
- Max Number of Objective Evaluations:** Input field contains '∞'.
- Target Objective Value:** Input field contains '-∞'.
- Max Run Time:** Input field contains '∞' and a dropdown menu is set to 'min'.

Figure 45: Optimizer Settings

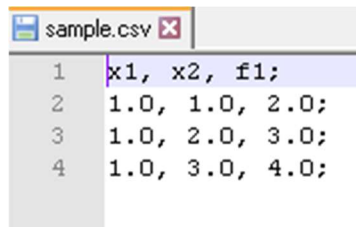
Table 3: Optimizer Settings Descriptions

Setting	Description
Number of Runs	The number of runs is the number of times to consecutively perform the optimization. Since the algorithms are stochastic, the results may differ on each run. Multiple runs can verify if the results are consistent.
Max Number of Objective Evaluations	The maximum number of function evaluations before the optimization is terminated.
Target Objective Value	The minimum objective value target for the optimization. The optimizer should terminate when it reaches this value.
Max Run Time	The maximum allotted time to run the optimization before stopping a run. Typing 0 will allow for unlimited time.

9.2 Import Points to Current Configuration

Design points can be imported from a file to enable the use of existing design points (e.g. known design data) for optimization and other analysis tasks. Before importing design points, the variables, objectives, constraints, and simulation integrations must be configured. To import design points, the following steps need to be followed:

1. Create a CSV (Comma Separated Values) formatted file with the design points. Data exported from a previous optimization run can be used if the variables, objectives, and constraints of the design points align with the active problem configuration.



```
sample.csv
1 x1, x2, f1;
2 1.0, 1.0, 2.0;
3 1.0, 2.0, 3.0;
4 1.0, 3.0, 4.0;
```

Figure 46: Example CSV file

2. The CSV file can be imported from the *File* menu with the *Import Points List...* option. If the points are imported successfully the user will be able to *Visualize* the results.

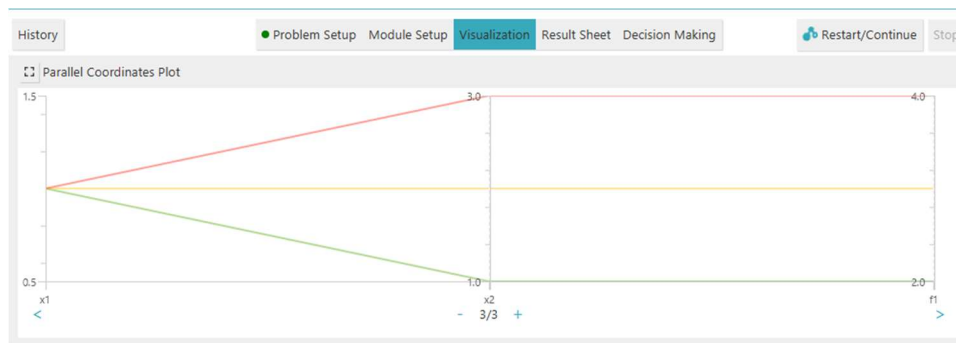


Figure 47: Visualization of Imported Points

The user can now either choose to continue the Optimization with the imported points or can start a new Optimization.

9.3 Export Optimization Data to Decision Making

Multi-objective optimization results can be exported to the *Decision Making* module. *Decision Making* can apply weights and sensitivities to the multi-objective data so that optimal designs can be identified.

To open the multi-objective optimization results in *Decision Making*, click on *Open in Decision Making* button in the *Result Sheet*. This will export all *Pareto Frontier* design points into *Decision Making*.

Open in Decision Making		Rerun Selected Points					
	Point No.	Pareto Frontier Me...	Feasible	f1	f2	f3	x1
<input type="checkbox"/>	167	yes	yes	20.96891294275103	23.978998543424602	46.038760176828326	0.5400452729966942
<input type="checkbox"/>	166	no	yes	19.138177197468693	31.57492730169635	45.060432775162475	3.171790085147517
<input type="checkbox"/>	165	no	yes	19.95919613427402	29.31822327093589	46.69562896634015	2.5167935159785095
<input type="checkbox"/>	164	yes	yes	20.566351408913206	31.026781347936975	40.374235644591444	1.9653676932620345
<input type="checkbox"/>	163	yes	yes	21.254481950479033	30.690303119506794	34.45929765674817	0.3711274144922516
<input type="checkbox"/>	162	no	yes	20.943341314218305	24.453116367077005	46.34087187128184	1.086909222900168
<input type="checkbox"/>	161	yes	yes	20.964492480903026	30.899832798338224	33.72629003990491	0.2664516816657492
<input type="checkbox"/>	160	no	yes	18.969842480213096	31.415338129336405	45.27101539060459	3.2408578665534877
<input type="checkbox"/>	159	yes	yes	18.6896206277119	30.910048912532694	45.998943033904915	3.2446914649008516
<input type="checkbox"/>	158	yes	yes	20.43033793615829	26.76998942918331	46.91152660259631	2.013327418984847
<input type="checkbox"/>	157	yes	yes	18.973053950444204	30.939011370649922	45.965133103352635	3.0433028590189224
<input type="checkbox"/>	156	yes	yes	20.29144091461803	31.737159258955625	40.94529810470003	2.2047504841322443
<input type="checkbox"/>	155	yes	yes	21.48385504750594	30.37466098299642	35.363556116772095	0.15386920621016098
<input type="checkbox"/>	154	no	yes	21.358106277120395	24.445447712519805	45.682021409686065	0.5236229844322743
<input type="checkbox"/>	153	no	yes	20.328976847357033	31.40784483159075	42.036215707214275	2.2359518635771125
<input type="checkbox"/>	152	yes	yes	21.53755258063455	29.38988433982947	40.020559827110176	1.2410083614591931
<input type="checkbox"/>	151	yes	yes	21.928172633901468	26.954564484977325	43.162201614858226	0.7685071050752967

Figure 48: Open in Decision Making

Load Design from file...
Back
Proceed to Rating...

Screening

Design Table

Design	f1	f2	f3
Point-7	21.1338409...	30.7260546...	38.0224796...
Point-11	21.4414008...	26.2445155...	44.8025578...
Point-17	21.4829293...	28.2812511...	42.5198846...
Point-19	19.4357666...	30.9713389...	44.1944436...
Point-21	20.5306595...	27.9501253...	45.3761567...
Point-22	21.4556895...	24.7419665...	45.0578804...
Point-24	21.8395056...	29.6622680...	37.4796750...
Point-25	19.7200070...	29.5924366...	45.2677697...
Point-28	19.2329930...	29.3472812...	46.7522358...
Point-29	21.5588439...	29.0510082...	41.1576700...
Point-30	21.9887047...	27.9328779...	41.0728422...
Point-33	21.5972853...	30.4811355...	35.8547421...
Point-35	19.0812887...	29.7723791...	46.3875667...
Point-36	20.2451314...	26.7004991...	46.8943879...
Point-37	20.2217923...	28.1721825...	46.1923398...

Configuration

Screening Attribute Setting

f1	18.68	22.10
f2	23.75	32.05
f3	33.88	47.09

Scatter Surface Plot

Figure 49: Imported Optimization Data from Results Sheet

53

10 Design of Experiments

The *Design of Experiments* (DOE) module enables the generation of sampling data using different DOE methods via an easy-to-use graphical interface.

10.1 DOE algorithms:

10.1.1 Latin Hypercube Design (LHD)

Generate an optimal Latin hypercube-based sampling scheme.

Settings:

- Number of points: Number of designs that will be generated.
- Iteration: Number of iterations to run for.
- Include Edge: Indicates whether the sampling points can be resided on the bounds of the design space.
- Random perturbation: Whether sampling points generated by LHD should be randomly perturbed.
- Use seed: Custom supplied seed to use for random point generation. When not supplied, OASIS will use the default random point mechanism.

10.1.2 Full Factorial Design (FFD)

Generate a full factorial sampling scheme.

Settings:

- Level: Number of levels for each variable. This setting determines the number of sample points generated. To set a level for a variable, select the variable from the list and you can edit the level for that variable in a text field (as shown in Figure 50). Default value is 1.
- Include Edge: Whether the sample points can be resided on the bounds of the design space
- Use seed: Custom supplied seed to use for random point generation. When not supplied, OASIS will use the default random point mechanism.

5. Click on *Evaluate* to begin generating objective and constraint values for the sampling data. If there are any issues with the current problem setup, a window will popup to inform you. Once all the problems are resolved, click on the *Evaluate* button and the evaluation process will start.
6. You can view the progress of the evaluated points in the *Visualization* and *Result Sheet*.

Open in Decision Making		Rerun Selected Points						
Point No.	Optimum	Feasible	30DPURT113	x1	x2	x3	x4	
<input type="checkbox"/> 547	no	yes	5.832315003085359E12	0.20588072692036974	-1.1969956069926893	-0.17404244923667056	-0.20544332287399	
<input type="checkbox"/> 546	no	yes	7.265438143503852E15	0.20588072692036974	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input type="checkbox"/> 545	no	yes	8.395608216263053E13	2.502938903751682	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input type="checkbox"/> 544	no	yes	2.547820214413181E14	2.8866219419812973	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input type="checkbox"/> 543	no	yes	2.7570132088292688...	2.928510795311719	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input type="checkbox"/> 542	no	yes	8.827949891565845E13	2.6227622907305	-1.9304917944091406	0.33080701395803214	-0.20544332287399	
<input type="checkbox"/> 541	no	yes	1.8209253622029406...	-0.7064012107644757	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input type="checkbox"/> 540	no	yes	5.450459473879975E13	0.20588072692036974	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input type="checkbox"/> 539	no	yes	2.217962726859388E15	0.20588072692036974	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input type="checkbox"/> 538	no	yes	9.322967666539295E14	0.20588072692036974	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input type="checkbox"/> 537	no	yes	1.3345521108141744...	0.20588072692036974	-0.9532312397028617	0.053597999126064666	-0.20544332287399	
<input type="checkbox"/> 536	no	yes	8.991834540983886E13	0.20588072692036974	-0.9532312397028617	0.053597999126064666	-0.20544332287399	
<input type="checkbox"/> 535	no	yes	2.154128274406987E12	0.20588072692036974	-0.9532312397028617	0.053597999126064666	-0.13429314941863	
<input type="checkbox"/> 534	no	yes	1.431673413542068E12	0.20588072692036974	-0.9532312397028617	0.053597999126064666	-0.20544332287399	
<input type="checkbox"/> 533	no	yes	2.9749647283642573...	0.20588072692036974	-1.79609087746019	0.053597999126064666	-0.20544332287399	

Figure 52: Results for Design of Experiment Run

Postprocessing

11 Results and Visualization

Results in OASIS are shown in three ways. First, data plots in the *Visualization* tab can help users visualize the optimization performance. Next, a “snapshot” *Run Status* is updated with numeric values for optimization metrics. Finally, the results can be saved to a Microsoft Excel format using the *File* → *Export Points to Excel*.

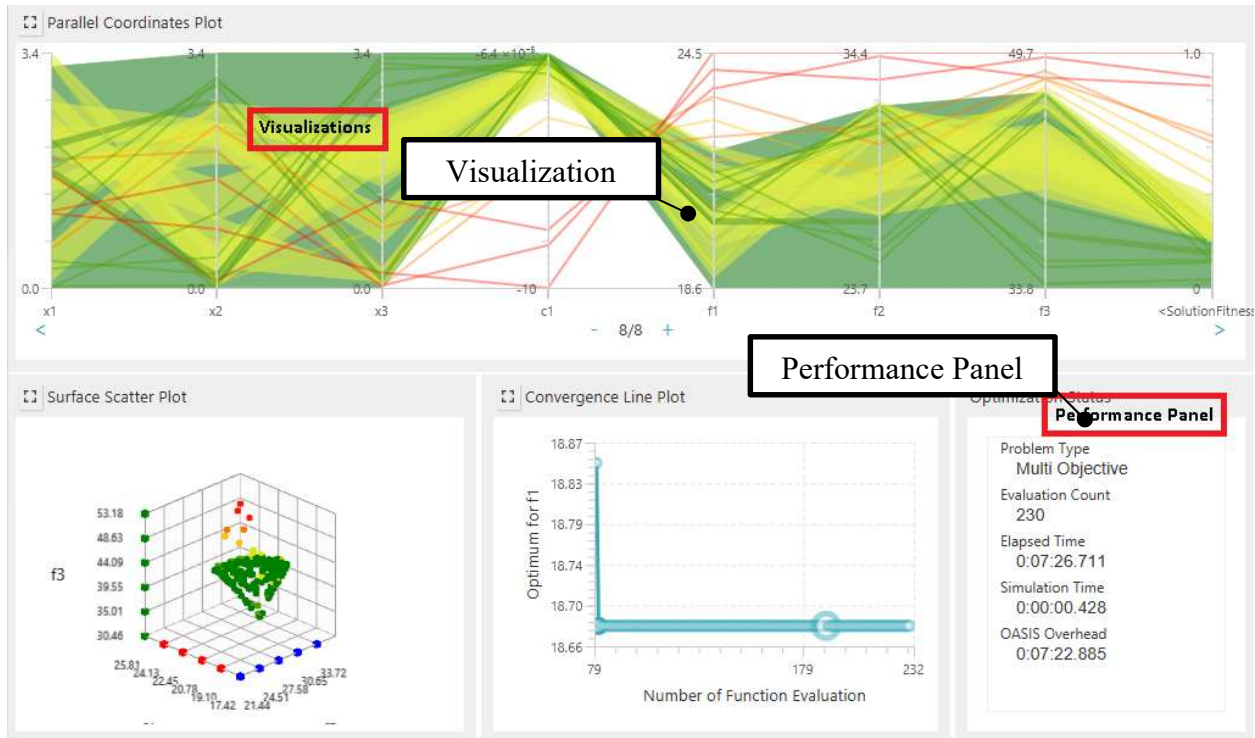


Figure 53: Results Display

11.1 Parallel Coordinates Plot

The graph in the top half of the *Visualization* tab is the *Parallel Coordinates Plot (PCP)*. PCP is a leading method for visualising multivariate data. In parallel coordinates, each variable is represented by its own axis. However, in contrast to a Cartesian plane, axes are laid out parallel to each other, allowing for many axes to be displayed simultaneously. In parallel coordinates, each data point is represented by a line that intersects the axes. The point of intersection is determined by the data point's value for that variable.

11.1.1 Colored Point Bands

Each of the coloured bands represents a group of points that have similar objective function values with red indicating poor designs, green indicating good designs with low objective function value, and light blue indicating infeasible designs. As the bands turn to light green, yellow, brown, and red, the function values of these groups of points are higher and thus less attractive from the perspective of optimization.

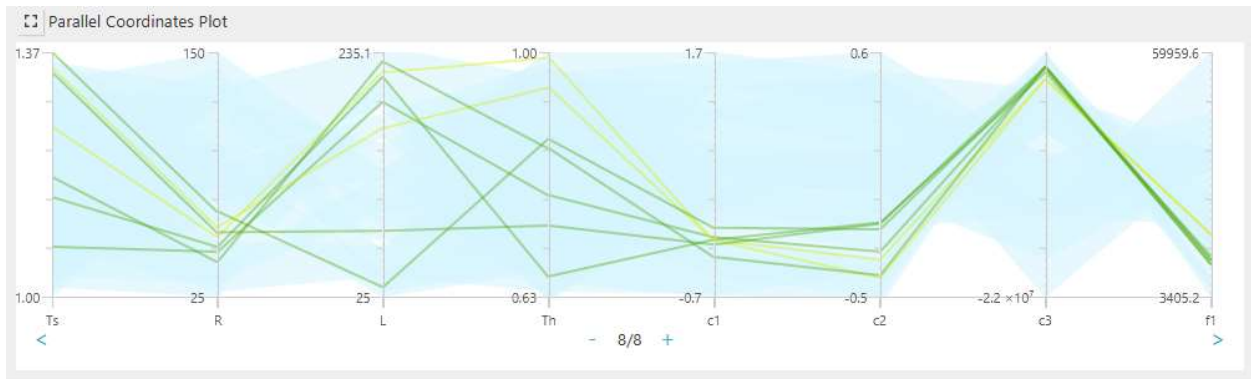


Figure 54: Infeasible (light blue), Moderate (yellow), and Good (green) Function Values

11.1.2 PCP Scrolling and Zooming

The number of axes displayed as well as which axes are displayed can be adjusted with a series of hotkeys, buttons, or the use of a right click dropdown menu. Figure below shows the hotkeys for those actions.

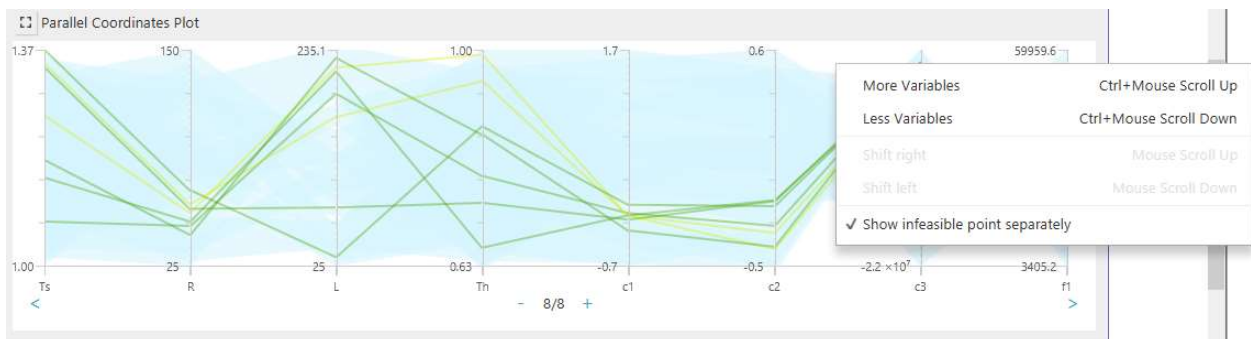


Figure 55: PCP Right-Click Menu

Figure 56 shows the buttons with which the same actions can be accomplished. The arrow buttons are for scrolling, and the “+/-” buttons are for zooming.



Figure 56: Parallel Coordinates Plot Zoom and Scroll Buttons

11.1.3 Fitness at the Right

In the PCP, the right-most axis always represents the fitness and constraint satisfaction of the points. In single objective problems, the fitness is represented fully by the objective function. In multi-objective problems, there is an additional artificial axis called *SolutionFitness*. The solution fitness represents the closeness of points to the *Pareto Frontier* (see Section 11.2.5)

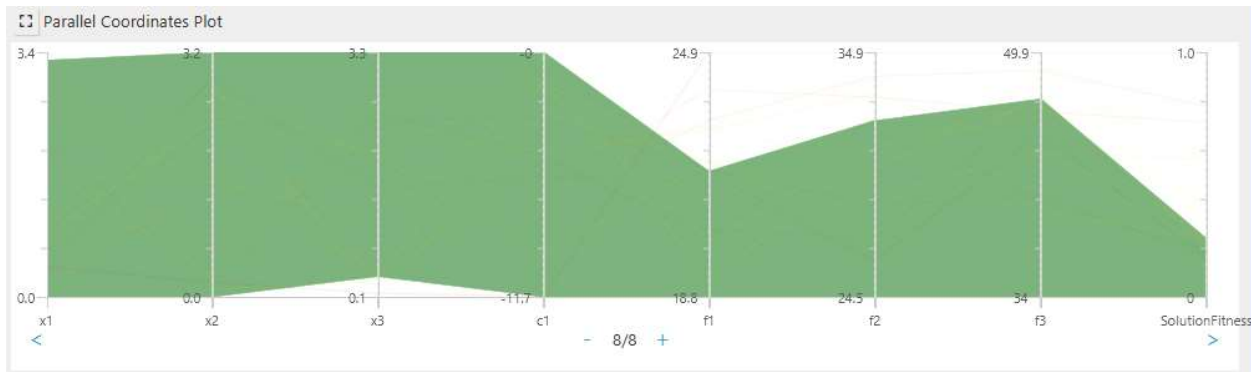


Figure 57: Solution Fitness and Pareto Frontier

11.2 Surface Scatter Plot (Single-Objective/ Multi-Objective)

The *Surface Scatter Plot* (the bottom-left pane) will plot any three input/output variables in a 3D space. The graph has different default settings for single-objective and multi-objective optimizations. A user is also able to customize each axis and choose which *Input* or *Output* to display for different analytical scenarios. *Surface Scatter Plots* (SSP) are in 3D; a user can rotate the graph and zoom in/out for better viewing of interesting data points or regions using the right mouse button or hotkeys as shown in the menu.

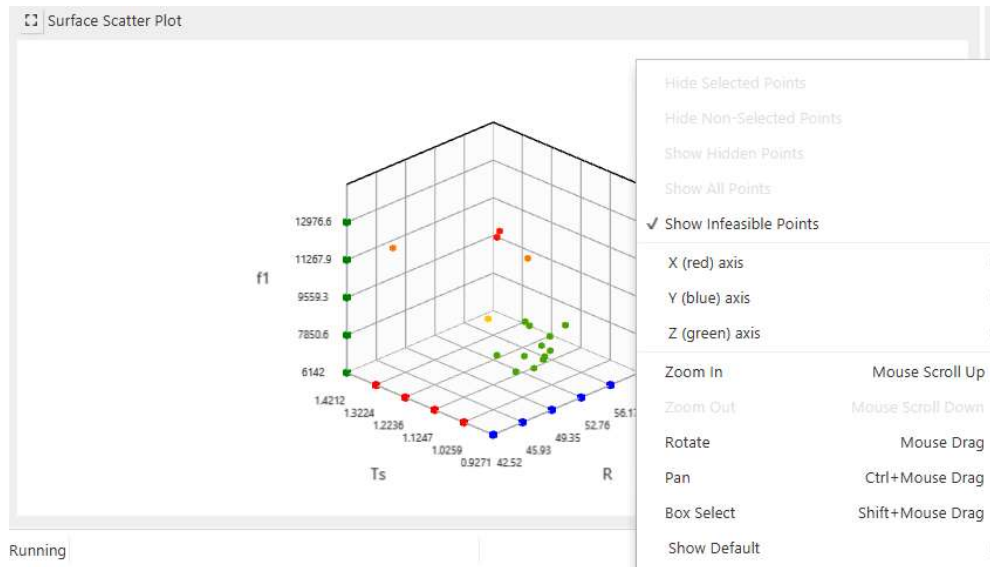


Figure 58: SSP with Right Mouse Button Menus

11.2.1 Selection of Points

The user can select any point inside the graph as a point of interest and this will be used for zooming. The selected point(s) will be moved to the center of the SSP so they are always in view. Once the point is selected, the point will be highlighted. To remove the highlighted point, click away.

- *Single Selection:* click on any point and the point will be highlighted in a different color
- *Multiple Selection:* hold the SHIFT key and then clicking on the desired points will allow multiple points to be selected. Shift clicking on a point already highlighted will remove it from the selection
- *Area Selection:* hold the SHIFT key then dragging with the Left Mouse Click will allow selection of all points in the rectangular region. To deselect, click away on the graph.

11.2.2 Detailed Point Information

The user can view the information of any point displayed on the graph by hovering the mouse over the desired point. An info panel will appear over the cursor. The info panel will disappear or change when you move away or move to another point.

11.2.3 Single-Objective Problem

For single-objective problems, the SSP will display the points with the *Objective* on the vertical axis and two *Inputs* on the other axes by default. The points are coloured in the same way as in the PCP (refer to Section 11.1).

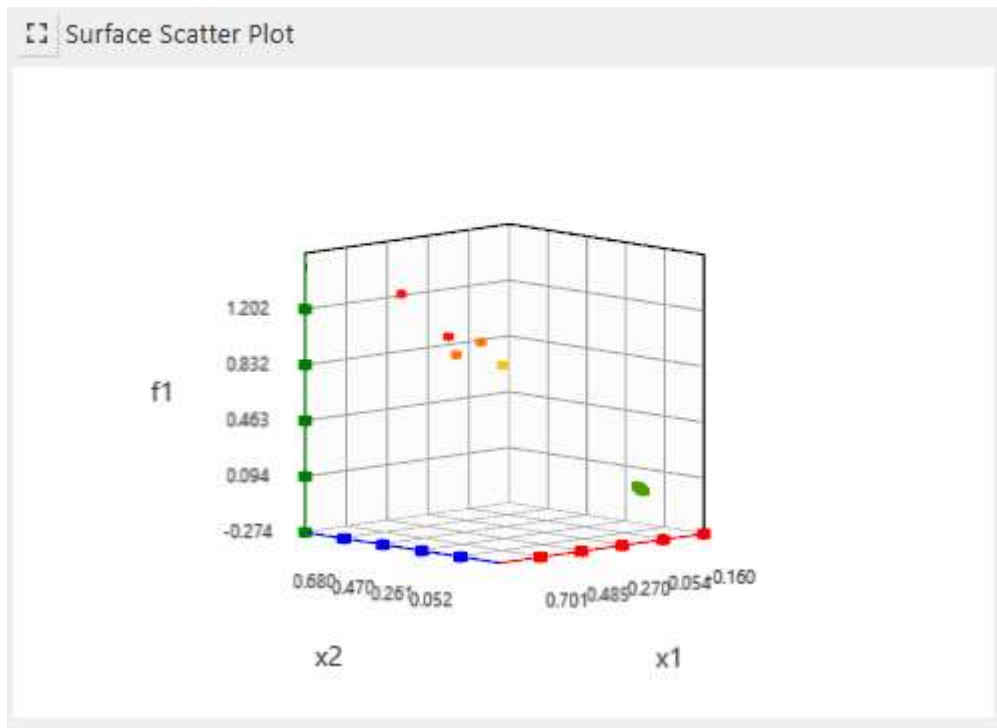


Figure 59: SSP for Single Objective

To change the input variable/function displayed on an axis, Right Click to bring up the menu and select the available options in the sub-menu of each axis.

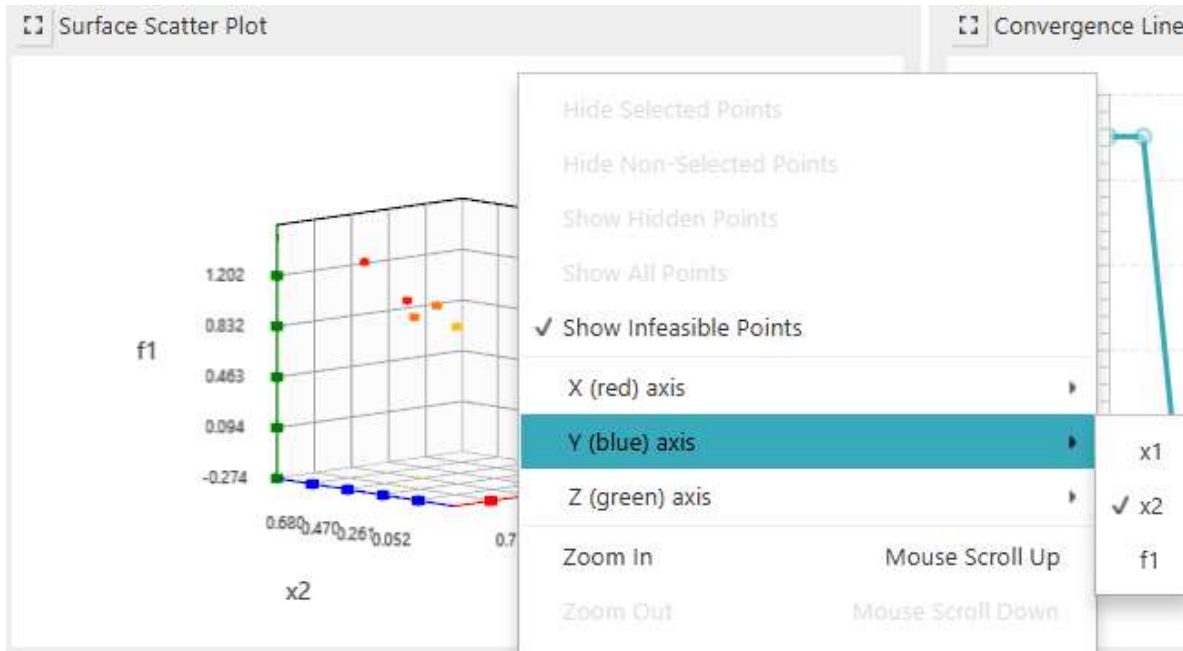


Figure 60: Axis Selection for SSP

11.2.4 Multi-Objective Problem

For multi-objective problems, SSP will display only the objective functions on the axes by default. These are known as *Pareto Frontier Plots*. For more information on the *Pareto Frontier*, see Section 11.2.5.

11.2.5 Pareto Frontier

The Pareto frontier (a.k.a. Pareto set or Pareto front) is the set of points that represent a “frontier” of best solutions for multi-objective problems and is the set of feasible points which are not “strictly dominated” by any other point. This means there is no other point that is better for *all* design criteria. Unlike a single-objective problem, multi-objective problems generally do not have a single best solution as the various objectives tend to conflict. If there exists a single point which is best for all criteria, that point is an elusive Utopia point. A Utopia point is rarely feasible and indicates that each objective could have been optimized individually (with single objective optimization). Therefore, the Pareto set is a commonly accepted solution set for multi-objective design problems with competing objectives.

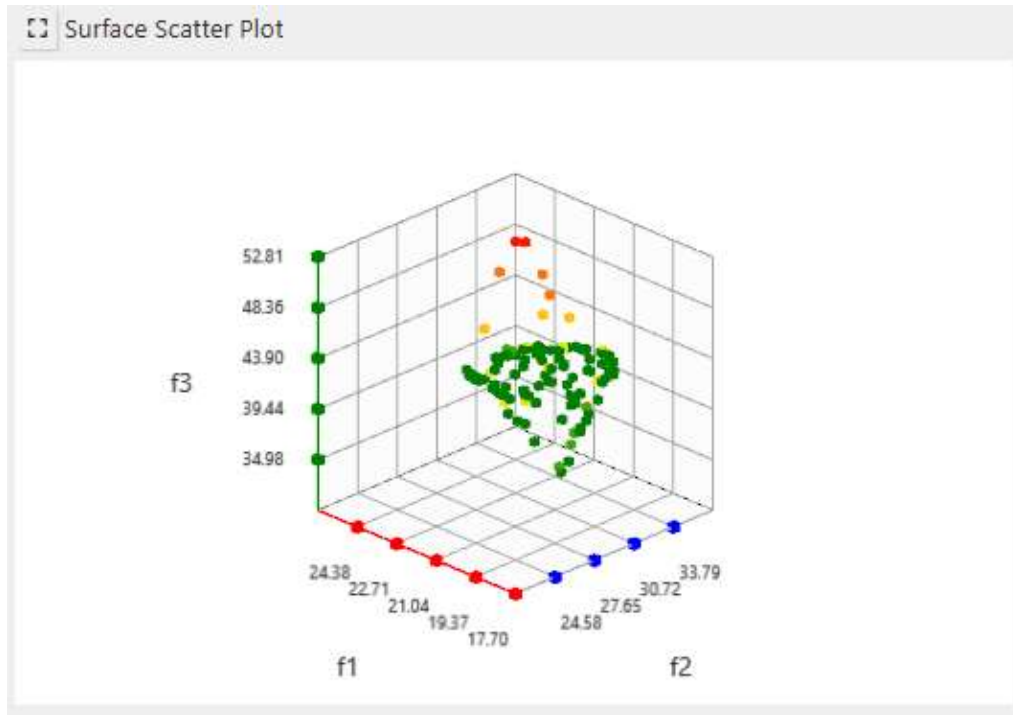


Figure 61: Pareto-set Plot

To show how the solution space expands over time, the Pareto set is plotted iteratively (Figure 61). Dark green points represent the Pareto frontier.

11.3 Convergence Line Plot

Another way to show the progress of the algorithm is to simply plot the optimal objective values over time. This type of plot is called the *Convergence Line Plot*. For both single-objective and multi-objective problems, only one objective function's progress will be shown. For multi-objective problems, you can switch the vertical axis to another objective function by using the right click menu. In the plot, a point that improved the current best optimum value will have solid color and bigger size. Circles with lighter color and smaller size represent function evaluations that did not improve the displayed objective. Function evaluations that contain invalid values will be highlighted in red.

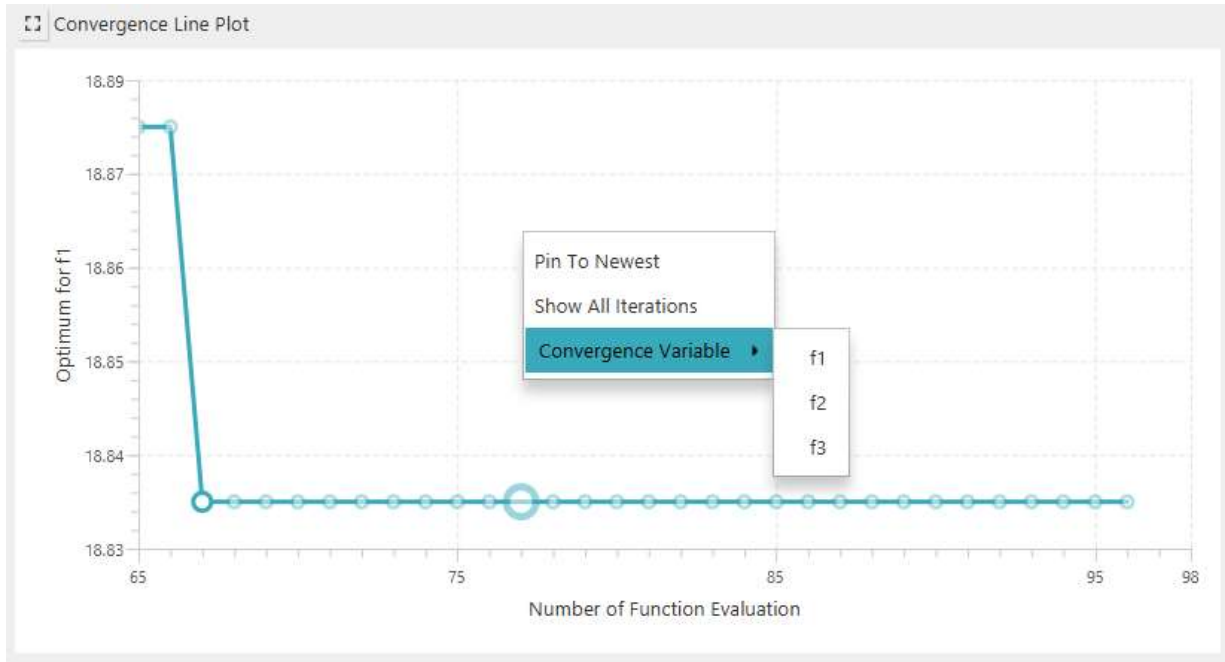



Figure 62: Convergence Line Plot

The *Convergence Line Plot* will update per iteration. To view the detailed information of a point, hover the mouse over it. The user can click on a point to pin the graph to a fixed location for detailed viewing. Clicking on the graph again will unpin the point and the graph will resume. The user can also zoom into a pinned or highlighted point. Use mouse scroll to zoom in or zoom out.

11.4 Full Screen Visualization

Each visualization can be expanded to occupy the full screen. To do so, click on the  icon located next to the visualization title. The visualization will then be maximized and the rest of the visualizations will be hidden.

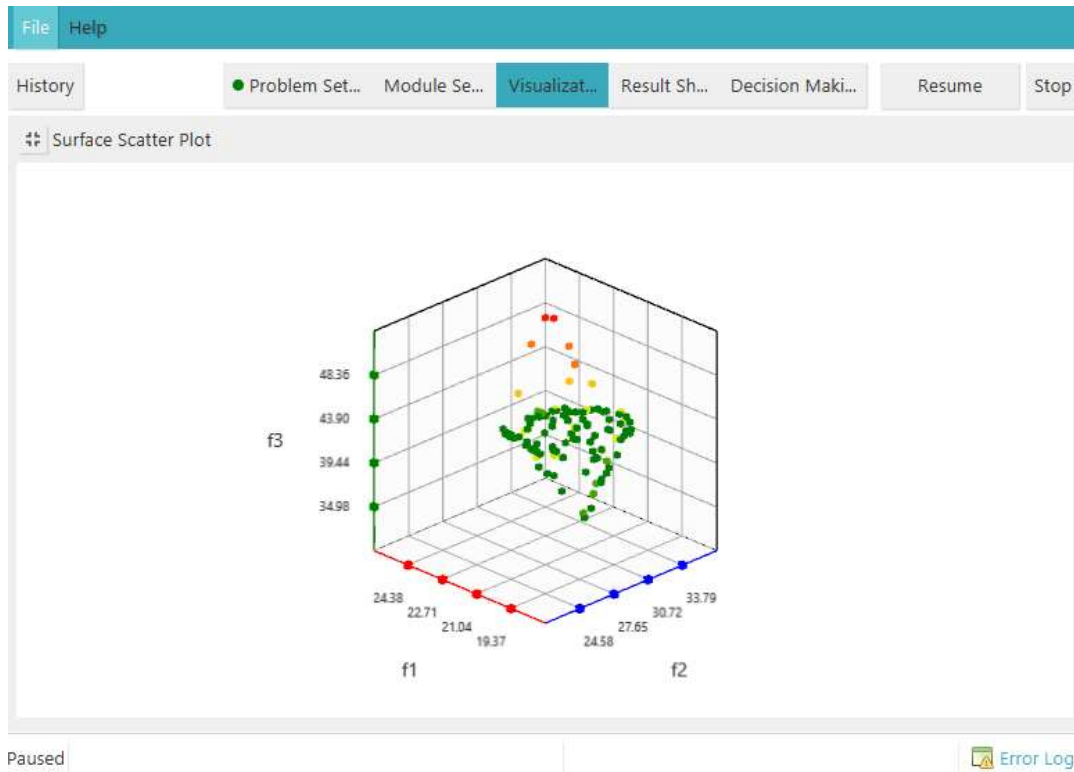


Figure 63: Full screen SSP


To exit full screen view of the visualization, click on the  icon. This feature is supported for the surface scatter plot, parallel coordinates plot and convergence line plot.



Figure 64: Full Screen CLP

11.5 Performance Panel

The *Run Status* pane provides a quick snapshot of the optimization progress with numeric values. The fields listed are provided in Table 4 below.

Table 4: Single Objective Performance Panel Fields

Performance Field	Description
Problem Type	The type of problem that is being optimized. There is <i>Single/Multi-Objective, Design of Experiment</i> or <i>Design Verification</i>
Evaluation Count	The total number of times the objective function(s) is evaluated.
Current Run Time	The total time elapsed while running the optimization.
Session Time	Overall time spent optimizing.
Simulation Time	Overall time spent running simulations.
OASIS Overhead	The coordinates of the point which generate the lowest objective function value.
Optimum Value	The lowest objective function value.

11.6 Excel Output of Results

Aside from the results shown on screen, the optimization data can be output to Excel using the *File* → *Export Results to Excel...* option. Results will be stored in the format shown in Table 5.

Table 5: File Output Block

x1	x2	Objective Function
1.51	0.02	2.18
0.61	0.47	0.80
0.42	0.60	-0.04
0.27	0.63	-0.49
0.17	0.66	-0.76
0.10	0.67	-0.88
0.05	0.68	-0.95
0.00	0.72	-1.00
-0.05	0.68	-1.02
-0.05	0.69	-1.02
-0.05	0.70	-1.02
-0.06	0.70	-1.03
-0.08	0.70	-1.03
-0.08	0.70	-1.03

11.7 Rerunning Points

The *Result Sheet* provides a means of rerunning points in the event of an unexpected output or failure. This can be done by selecting any number of points in the *Result Sheet* and then clicking the *Rerun Selected Points* button.

		Open in Decision Making		Rerun Selected Points					
	Point No.	Optimum	Feasible	30DPURT113	x1	x2	x3	x4	
<input type="checkbox"/>	547	no	yes	5.832315003085359E12	0.20588072692036974	-1.1969956069926893	-0.17404244923667056	-0.20544332287399	
<input type="checkbox"/>	546	no	yes	7.265438143503852E15	0.20588072692036974	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input checked="" type="checkbox"/>	545	no	yes	8.395608216263053E13	2.502938903751682	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input checked="" type="checkbox"/>	544	no	yes	2.547820214413181E14	2.8866219419812973	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input checked="" type="checkbox"/>	543	no	yes	2.7570132088292688...	2.928510795311719	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input checked="" type="checkbox"/>	542	no	yes	8.827949891565845E13	2.6227622907305	-1.9304917944091406	0.33080701395803214	-0.20544332287399	
<input checked="" type="checkbox"/>	541	no	yes	1.8209253622029406...	-0.7064012107644757	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input type="checkbox"/>	540	no	yes	5.450459473879975E13	0.20588072692036974	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input type="checkbox"/>	539	no	yes	2.217962726859388E15	0.20588072692036974	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input type="checkbox"/>	538	no	yes	9.322967666539295E14	0.20588072692036974	-1.1969956069926893	0.33080701395803214	-0.20544332287399	
<input type="checkbox"/>	537	no	yes	1.3345521108141744...	0.20588072692036974	-0.9532312397028617	0.053597999126064666	-0.20544332287399	
<input type="checkbox"/>	536	no	yes	8.991834540983886E13	0.20588072692036974	-0.9532312397028617	0.053597999126064666	-0.20544332287399	
<input type="checkbox"/>	535	no	yes	2.154128274406987E12	0.20588072692036974	-0.9532312397028617	0.053597999126064666	-0.13429314941863	
<input type="checkbox"/>	534	no	yes	1.431673413542068E12	0.20588072692036974	-0.9532312397028617	0.053597999126064666	-0.20544332287399	
<input type="checkbox"/>	533	no	yes	2.9749647283642573...	0.20588072692036974	-1.79609087746019	0.053597999126064666	-0.20544332287399	

Figure 65: Rerunning Points

11.8 End Report for Optimizations

A detailed report of an optimization run appears when the run is over. This report is also accessible in the *Result Sheet*. The *Session Report* displays summarized information about the results. The information summarized in the report is:

- Number of iterations
- Total run time
- Total simulation time
- The best design and/or best designs (Pareto frontier)
- Performance metrics: objective value, start value, end value, and improvements in-between
- Best band: point numbers and value ranges

This detailed report is also exportable to Excel.

12 Decision Making

This module assists users to choose one from many suppliers for a specific component, or one concept from many to proceed to the detailed design stage. Most of time, making selection is compacted when the scale grows. Our decision making module helps us to deal with this by visualizing using parallel coordinate plot as visualization.

12.1 Glossary

12.1.1 Attribute

Attribute is the factor we compare between different suppliers/concepts in order to choose one over the other. For example, two suppliers A and B supply the same part for an assembly, the attributes can be the price, quality, or material. Generally quantifiable attributes can be easily compared. For instance, price can be quantified as a currency figure. Quality, as a subjective measure, needs to be rated in a scale. For material, unless we have a preferable metric, such as the strength, the name of the material itself might not be a good attribute in the selection process.

- Preference

For each attribute, one needs to know which value of the attribute is preferred. For example, lower cost and high quality are generally preferred. The user thus need to tell the selection module their preferences.

- Weight

The weights indicate the importance of each attribute. The higher the weight, the more important is the attribute. Weights can be added directly to an import document by adding a row with the name "[Weights]" and a corresponding weight value in each of the cells of that row.

12.1.2 Data

A user can manually enter the attribute data, such as the engine power, cost, efficiency, etc.

12.1.3 Rating

Rating is used to measure a supplier for a given attribute. It is an integer ranging from 1 to 100. The higher the value, the better is the performance. The rating can be automatically converted by OASIS from the raw data or can be entered by the user.

12.1.4 Sensitivity

Each rating can be assigned a sensitive value. With sensitivity, a rating can take a value range. This will enable users to apply uncertainty on a specific rating. By clicking on a *Rating* box, the *Rating Sensitivity* pane appears to enter the sensitivity information by percentage or absolute rating values.

12.2 Tools

12.2.1 Screening phase

The *Screening* tool is for reducing the number of Pareto points we proceed to rate. It's a way to weed out the worst of the best points before we move onto the in depth analysis.

12.2.1.1 Design table

A user is expected to have a raw Excel table that records the raw data or ratings for the chosen attributes. Once a *Decision Making* spreadsheet is opened in OASIS, the table should be filled. Now you can keep working on the data that in the spread sheet and any changes you made here will be reflected on the SSP and screening tool. However, the original spread sheet will not be changed.

Design Table

Design	Initial Cost	Maintenanc...	Cost scales ...	Feature A	Feature B	End custom...	Customer Su...
Company A	9000.0	1800.0	1.0	0.0	10.0	5.0	5.0
B Inc.	0.0	2500.0	1.0	0.0	9.0	6.0	10.0
Enterprise C	0.0	1800.0	1.0	0.0	7.0	7.0	8.0
D Corp.	4000.0	1000.0	0.0	8.0	8.0	7.0	7.0

Figure 66: Supplier Table

The *Design Table* works similarly to Excel. Select a cell, start typing the new value, then hit enter. The changed value will be recorded and reflected on the visualization plots.

12.2.1.2 SSP

SSP will plot in a 3D space defined by 3 chosen attributes, which can be toggled using the right click menu. SSP in *Decision Making* works similarly to the one in *Visualization*. You can rotate the graph, zoom in and out of a point or group of points, hiding points, etc. Using the right click menu will help you with most of actions you can perform on the SSP.

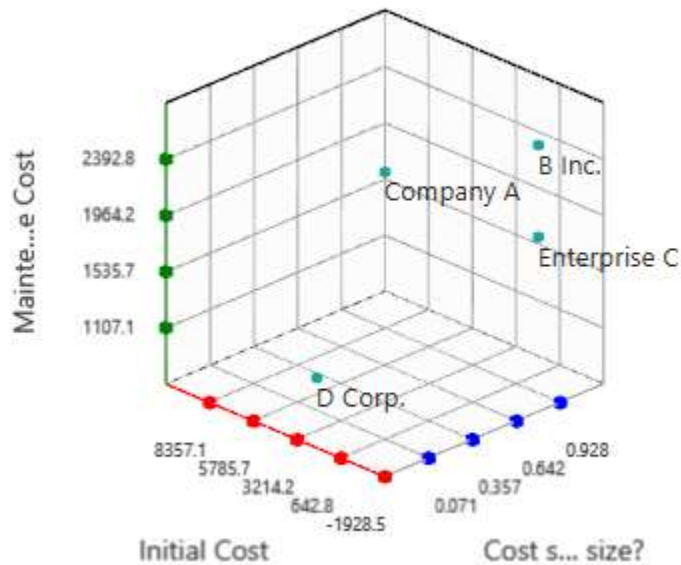


Figure 67: SSP Plot in Decision Making

12.2.1.3 Screening Tool

You can define the range that you are interested in for each attribute in the screening tool. You can input the value directly or you can use the slider bar to choose the range that you want. The screening will happen on-the-fly for SSP and the *Design Table*. The candidates satisfying the desired range for all attributes will stand out and the ones that failed the screening will be grayed out.

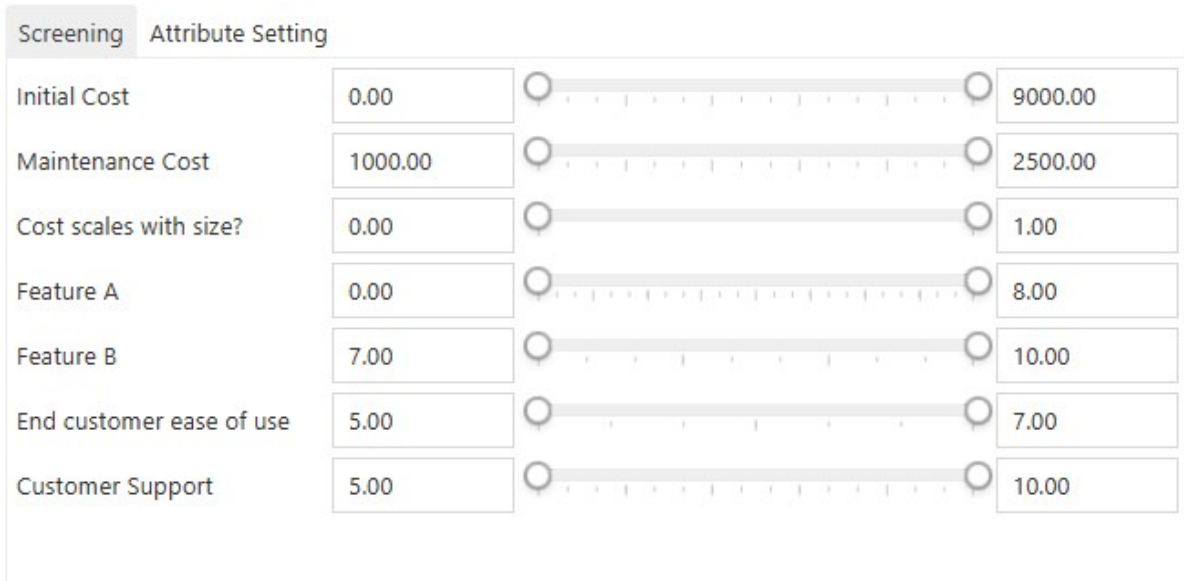


Figure 68: Screening Tool.

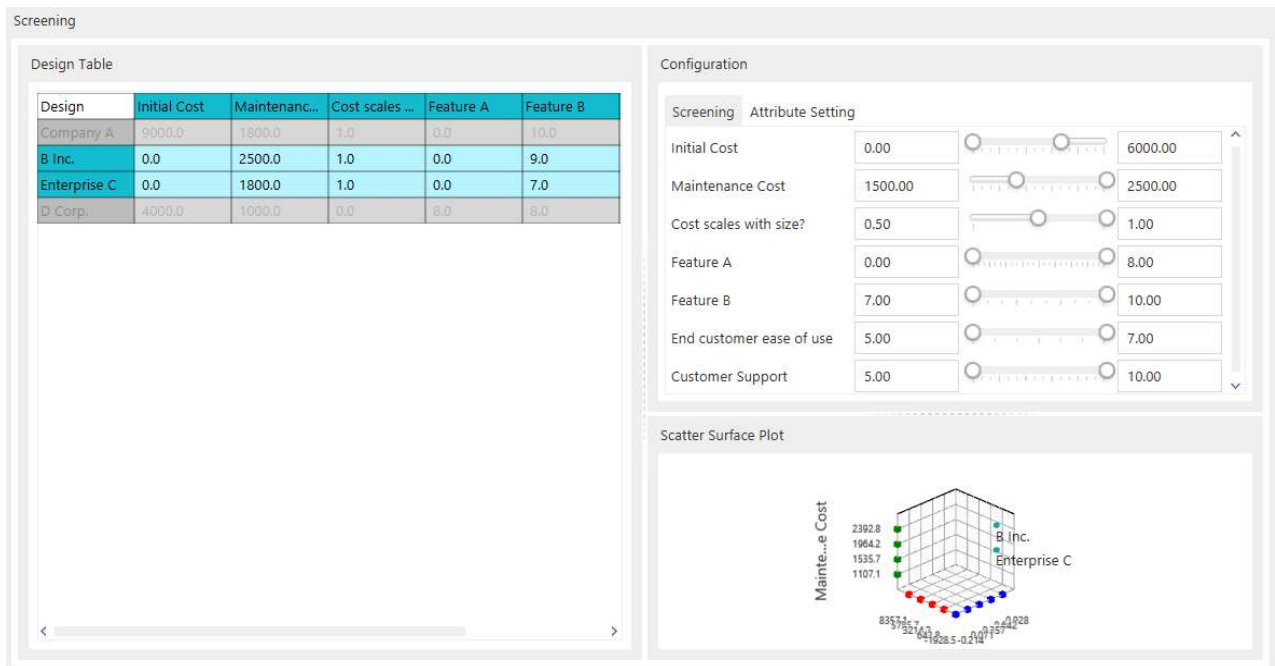


Figure 69: The GUI at the Screening Stage

12.2.1.4 Attribute setting

In the *Attribute Setting* window, you can input weights for attributes. The default is to assign equal weights to all attributes. It is also the place that you inform OASIS your preference of these attributes.

Attribute	Weight	Percentage	Preference
Initial Cost	1.0	14.29%	Prefer High Value
Maintenance C...	1.0	14.29%	Prefer High Value
Cost scales wit...	1.0	14.29%	Prefer High Value
Feature A	1.0	14.29%	Prefer High Value
Feature B	1.0	14.29%	Prefer High Value

Figure 70: Setting Attributes

You can modify the weights at both the *Screening* and *Selection* stages. However, you can only edit the preference (high or low value) in the *Screening* stage. This is because the auto rating feature will automatically rate the suppliers based on these preferences.

12.2.2 Proceed Options

After the Screening stage, one can choose to Proceed (the top menu). There are five options that one can choose to proceed the Selection stage. These options are explained in the dialogue window as shown below.

Continue without changes

No rating will be assigned and changed. This works most when you want to assign your own rating or you come back from screen phase after some screening changes.

Normalization

Automatically rate the supplier base on the minimum and maximum.

Customized normalization

Automatically rate the supplier base on the minimum and maximum provide by user.

Clear rating and sensitivity

Leave all the rating as default “1”.

Cancel

Go back.

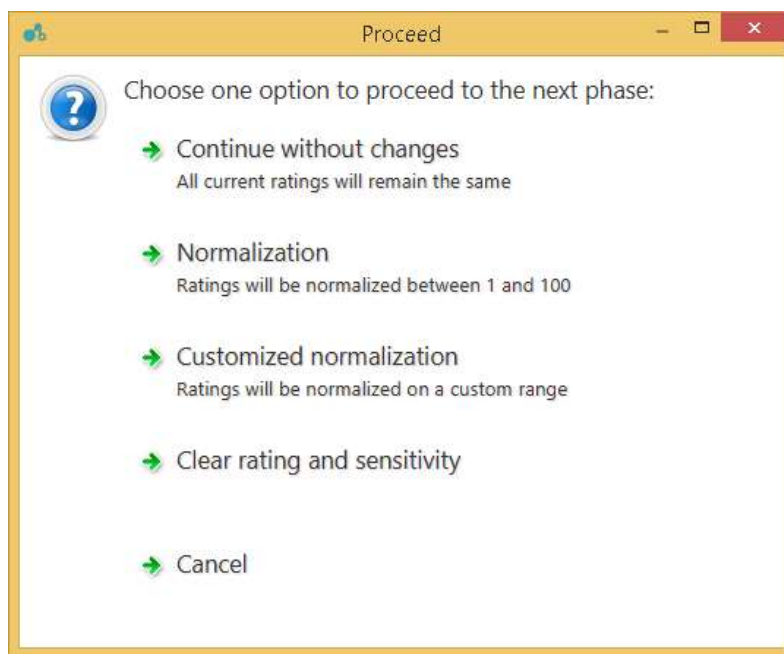


Figure 71: Options to Proceed to the Selection Stage

When you choose *Customized Normalization* mapping, another window appears so you can input the ranges you need. The attribute data with minimum value will be assigned the smallest rating and the one with maximum value will be assigned largest rating. By default, the range for each attribute is the minimum and maximum of all values in the same attribute.

Attribute	from	to
Ease of Handling	3.0	4.0
Ease of Use	3.0	4.0
Number Readability	2.0	5.0
Dose Metering	2.0	3.0
Load Handling	3.0	56.0
Manufacturing Ease	2.0	3.0
Portability	3.0	3.0

Figure 72: Customize Attribute Ranges

12.2.3 Result phase

12.2.3.1 Design Table

You can change the rating here and the result will be reflected on the PCP. Although the data fields are shown here, you are not able to change them; they are shown only for reference. When you click on the rating table, different configuration tools will pop up depending on the type of cells being selected.

12.2.3.2 Rating Sensitivity

The *Rating Sensitivity* tool will show up when you click on *Rating* underneath an attribute. When clicking on a single rating cell, you can input the sensitivity in the way you want, either using the actual rating number or a percentage value.



Figure 73: Multiple Rating Sensitivities

You can also select multiple cells by shift click on ratings to set the sensitivities altogether. Once you finish the change, the ratings that have sensitivity information will show as a cell with a red mark at the right bottom corner.

12.2.3.3 Weighting tool

When clicking on the attribute cell, the *Weight* tool will show up. This time you will not be able to setup preference since we have already used your preference settings from the *Screening* stage. Since we are at the *Selection* stage, any weight change will directly affect the selection result, which is shown in PCP as you change the weights.

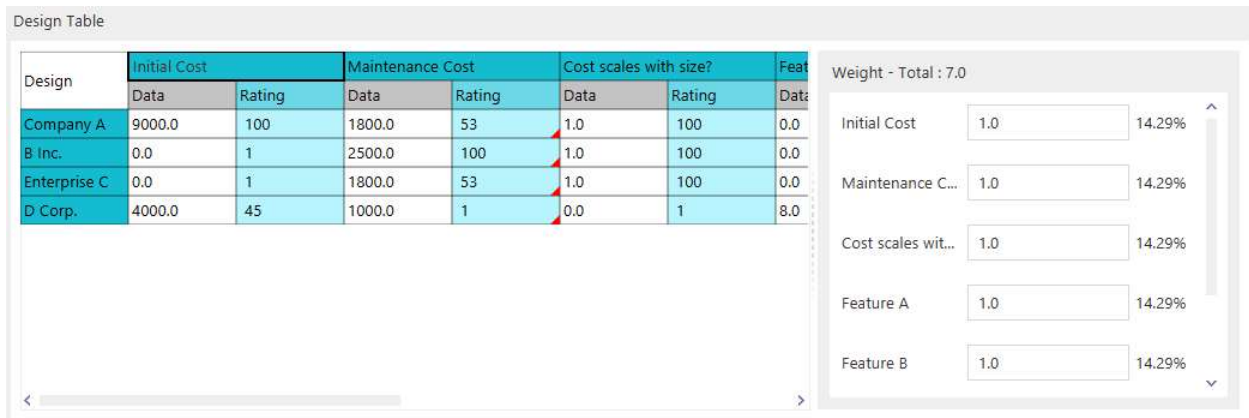


Figure 74: Adjusting Weight Selections

12.2.3.4 PCP

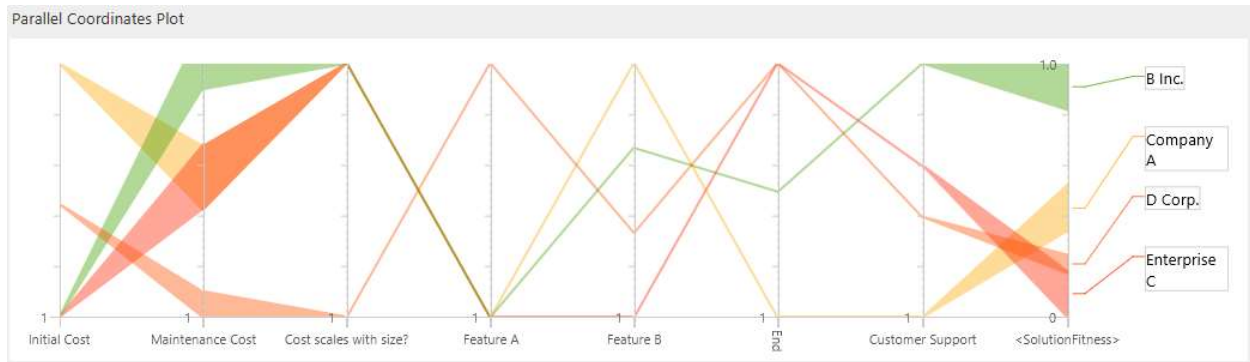


Figure 75: PCP in Decision Making

PCP reflects all the ratings and associated sensitivities. All the attributes will be evaluated based on the weights. The final fitness at the last axis to the right, i.e., the weighted sum ratings, shows the comparison between the options. The higher the fitness, the better the choice is. Hovering over the names or the PCP lines or bands will highlight the options to help you examine the data. You can also click on the line/band to make the option stand out to help the inspection. A line in PCP represents the attribute rating without sensitivity, and a band represents the attribute with sensitivity. Lines and bands are colour coded based on the fitness value. In general, green indicates the preferred and red indicates the least desired candidate.

12.3 Workflow

Here are steps to start a *Decision Making* project

1. Import data from a Multi-Objective optimization or manually create an Excel table.

One needs to organize the raw data with rows being the candidates and columns being the attributes, like Table 6. Remember, you can import data from Multi-Objective optimization runs, as stated in Section 9.3.

2. Load the data
3. Screening: Select the *Preference* and enter the weights if necessary
4. Proceed
5. Tune the *Rating* and add *Sensitivity*

One can fine tune the rating at this stage and add sensitivity information for your rating values.

6. Inspect on PCP

PCP plots the preferred candidates, which is also the final output of the Supplier/Concept module.

12.4 Case Study: Choosing Between a Software Product

This sample case study demonstrates the practical use of the *Decision Making* module. The problem we have is choosing a software product out of offerings from 8 companies. There are 7 attributes that we consider important in the evaluation and data is collected on each of the competing offerings, as shown in the table below.

Table 6: Evaluation Attributes and Collected Data

<i>Supplier</i>	<i>Initial Cost</i>	<i>Feature A</i>	<i>Feature B</i>	<i>Feature C</i>	<i>Feature D</i>	<i>Customer Support</i>	<i>Customer Support Cost/year</i>
Supplier A	5000	7	6	7	0	9	0
Supplier B	7000	7	4	7	0	7	0
Supplier C	9000	8	5	5	4	6	0
Supplier D	0	6	2	9	0	8	1000
Supplier E	0	8	2	0	4	5	1000
Supplier F	0	7	2	3	7	8	1000
Supplier G	5000	7	8	7	0	5	500
Supplier H	7000	8	10	0	5	5	700

The table above is first entered into an excel spreadsheet which is then loaded into the *Decision Making* module via the *Load Design from File...* button, and OASIS should look like the following figure.



Figure 76: Decision Making with Loaded Data

At this stage, data that is not wanted in the final analysis can be screened out. For example, suppose we have a budget of only \$8000 to spend on the initial purchasing price of the product. The slider value for the Initial Cost criterion can be adjusted to screen out all suppliers with initial cost greater than \$8000, as shown in the following figure.

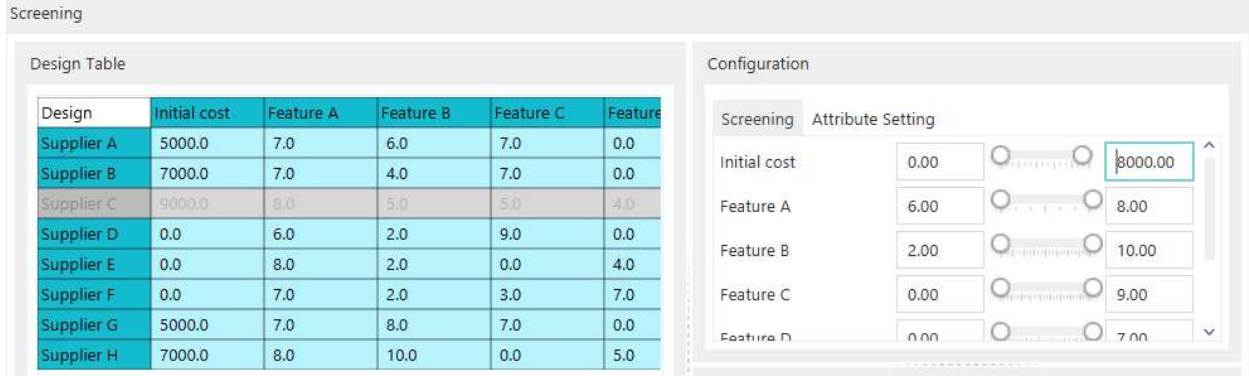


Figure 77: High Cost Supplier Screened Out

Aside from screening, it is also important to define the weight for each attribute as not all attributes have equal importance in our decision making. The attribute settings can be accessed from the “Attribute Setting” tab. Weight values can be entered for each attribute, and the percentage importance of each attribute is calculated automatically. In addition, the directionality of each attribute can be set using the *Prefer High Value* checkbox.

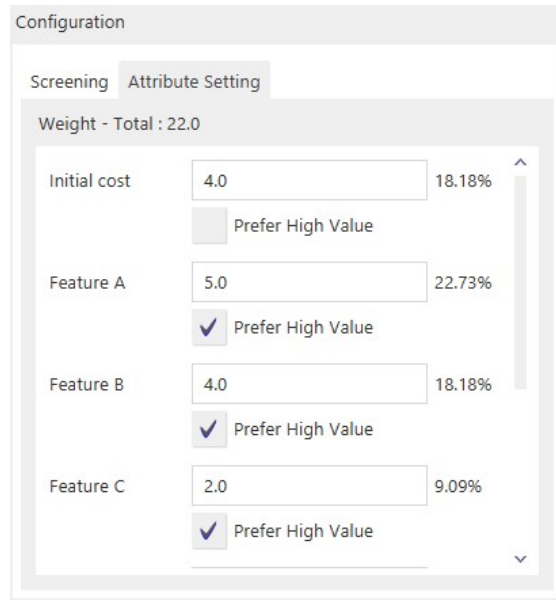


Figure 78: Attribute Settings

Now, we select *Proceed to Rating* to enter the second phase where we can perform further analysis to determine a ranking of the suppliers. The transition from the first phase to the second phase involve a mapping of the raw data imported in the first phase into ratings. We select *Customized Normalization* and enter the following values:

Category	from	to
Ease of Handling	3.0	4.0
Ease of Use	3.0	4.0
Number Readability	2.0	5.0
Dose Metering	2.0	3.0
Load Handling	3.0	56.0
Manufacturing Ease	2.0	3.0
Portability	3.0	3.0

Figure 79: Customized Mapping of Data to Ratings

These range values are used to calculate ratings that range between 1 and 100 from the raw data values. The following figure shows OASIS in the second phase:

OASIS Operation and Navigation

13 Project Structure and Navigation

In OASIS, a *Project* consists of one or more *Scenarios*. Each *Scenario* represents a different case or problem. OASIS incorporates a flexible and versatile project management system that enables you to create, save and load optimization configurations, decision making content, and entire project files.

13.1 Project Files

Projects can be created from the *File* menu. To create a project from the *File* menu, click *File* then *New Project*.

To load a previously saved project, click *File* then *Open Project...*, and then select the desired project file. OASIS project files have file type ‘.oasis.’ Alternatively, a user can click on *Continue* from the Start-Up window. OASIS will load the last project which was being worked on.

To load a recent project, select the file from the *Most Recent Project...* drop-down menu. If the project file no longer exists, the user will be prompted with an error message and the option will be removed from the list of most recent projects.

To save a project, click *File* then *Save*. To save the project as a new file, click *Save As...* instead.

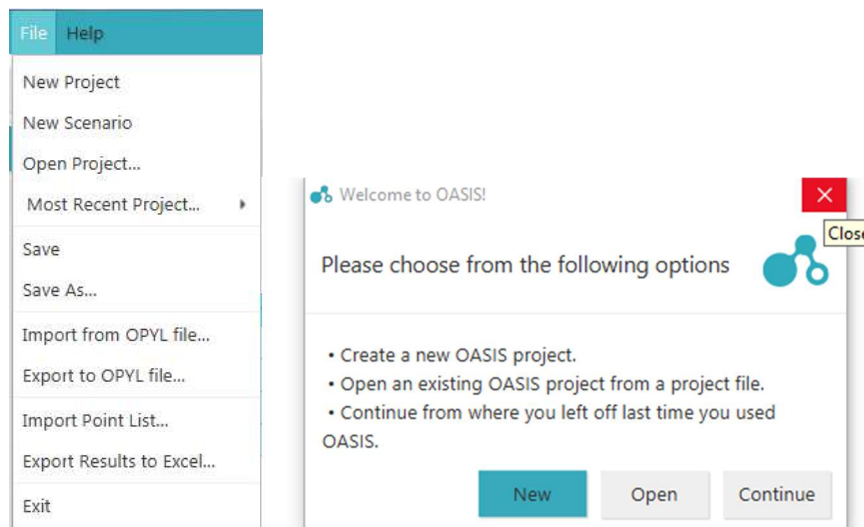


Figure 80: File Menu (Left) OASIS Start Up Dialog (Right)

13.2 Project History and Navigation Overview

The project *History* can be revealed by selecting the *History* button on the upper left of the main OASIS window. It shows all key elements and structure of the Project currently in use and allows navigation of Project elements.

The project *History* is subdivided into different scenarios. Each scenario is associated with a certain problem setup. Whenever a change is made to the problem configuration (e.g., changing a variable bound or any change in constraints or objectives), the system will automatically generate a message pertaining to the respective changes.

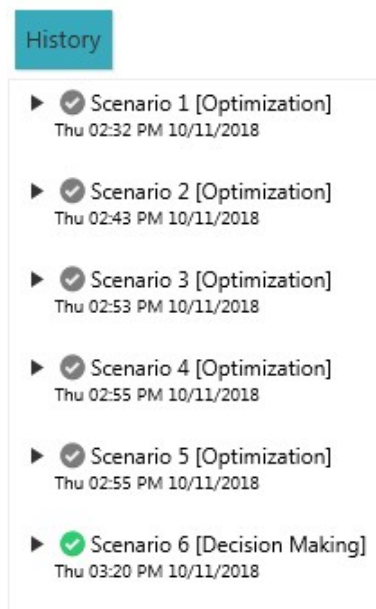



Figure 81: History Tab

The scenario with the green highlighted checkmark is the active scenario. All actions performed in OASIS such as running an optimization or importing design points are done in the active scenario. To move between the different scenarios, a user can double click the  symbol. Information pertaining to the scenario such as its configuration and relevant optimization runs are shown when the user expands the scenario by clicking on the arrow.

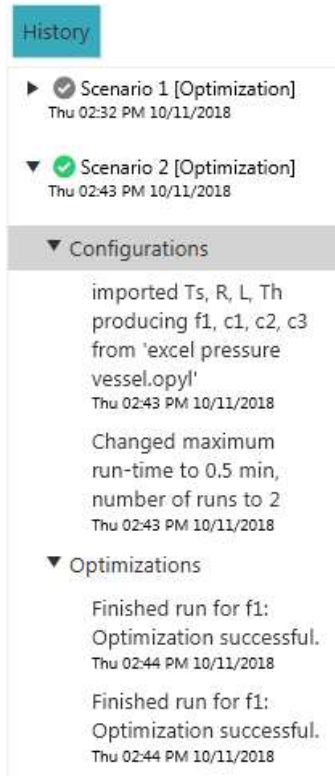


Figure 82: Expanded Scenario Example

By right clicking the *Optimizations* section in a scenario, the user can either *Reset configuration to here*, *Show Results* pertaining to that optimization run, or *Delete* the results from that run.

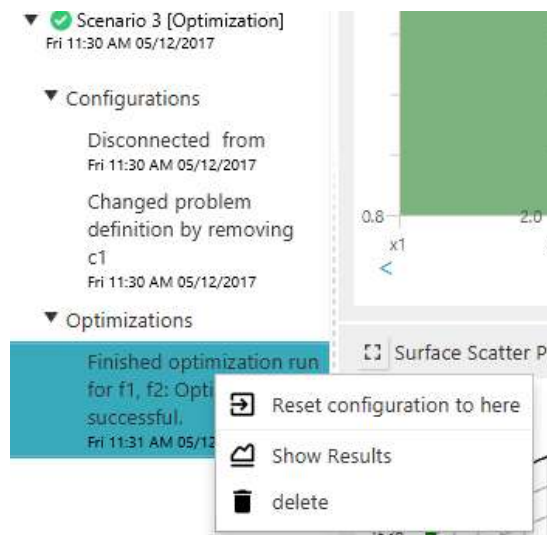


Figure 83: Right Click History Menu

14 OASIS Command Line

OASIS has a Command-Line Interface (CLI) that can be used to stream-line execution of projects that generate many points. In the graphical mode for mathematical problems OASIS spends a lot of resources updating the various visualizations and persistence mechanisms. In the CLI mode, the OASIS overhead is reduced significantly. The CLI mode is recommended for (but not restricted to) purely mathematical projects, and not recommended for simulation integration configurations. This is because even very fast external simulations that can be computed on the order of seconds will require significantly more time to execute than the OASIS visualization updates, meaning OASIS in the graphical mode will be nearly as fast as OASIS in the CLI mode.

14.1 Setting up a Project for Execution in CLI Mode

To run OASIS from the command line you must first run the regular graphical mode to create a project file that will be used to specify the configuration to run. This project file also serves as the location to store results.

Run OASIS in the graphical mode and save the finished configuration to a known location. The full path to the file, from the drive letter, to file extension, will become the `project path` variable for the CLI invocation.

One key difference between the CLI mode and graphical mode is that in the CLI mode the specified configuration must have a definite end. This means one of the convergence criteria must be set, such as maximum number of function evaluations or maximum runtime.

The CLI-mode OASIS cannot be stopped mid optimization. If the process is interrupted via a `ctrl + c` or other kill command, no results will be saved to the specified project file.

14.2 Running a Project File in Command Line

OASIS supports a standard help script, which can be accessed by changing to the installed OASIS directory, and running:

```
oasis.cli.exe --help
```

will give you the configuration options.

14.2.1 OASIS CLI Options & Arguments Table

Table 7: Options for the CLI Mode.

option	option longhand	description
-e	--execute <project path>	starts OASIS running the specified project file. Can be combined with '--headless' to be run in a pure command-line mode. Optional from GUI.
-h	--help	prints the summary [this table] about usage of OASIS
-l	--log <logging level>	specify the minimum level of message severity to be logged. Options are one of {OFF, SEVERE, WARNING, STATEFUL_EVENTS, EVENTS, INFO, CONFIG, FINE, FINER, FINEST, ALL}. Defaults to WARNING
-sp	--startupPreference <opening preference>	specifies which option to perform at startup, one of { LoadExistingProject, LoadMostRecentProject, CreateNewProject }
-v	--verbose	enable verbose logging (synonymous with -log FINE). Users should not use both 'verbosity' and 'log' options on the same invocation, as the value for 'log' will override 'verbose'
-x	--headless	runs OASIS in a non-graphical ('headless') mode

*Note that further customization of OASIS can be done by specifying **Error! Reference source not found.***

14.2.2 Example Calls

OASIS can be run according to

oasis.cli.exe "C:\Users\ThisUser\Documents\saved oasis project.oasis"
 or, more verbosely and using the regular executable:

```
oasis.exe --headless --execute "C:\Users\ThisUser\Desktop\Another Oasis Project.oasis"
```

The difference between `oasis.exe` and `oasis.cli.exe` is that `oasis.exe` respects windows GUI process conventions, creating a new process for our program, whereas the `oasis.cli.exe` respects standard terminal conventions, which involves execution under the calling processes and is hard-coded to run with the `--headless` flag.

14.3 Viewing the Results

Once the command line has finished executing the configuration specified in the project file, it will save the results to that project file. The specified project file can be opened using OASIS in the graphical mode with the results loaded in from the *History* tab.

15 Troubleshooting.

15.1 General Troubleshooting Tips

This section helps list some common issues our users face and tips on how to resolve the issues.

1. An external simulation does not yield any valid results (for example, OASIS throws a ‘no output found’ error).
 - a. The first step is to ensure the simulation integration is working properly. Although OASIS may think a simulation integration is set up properly and not show any errors on the *Optimize* button, there may still be something erroneous with the setup. We advise users to run their simulation independent of OASIS to confirm it works properly.
 - b. If an output file is being generated correctly in the above step, we suggest double-checking to make sure the file is in a readable format. Often, we find that simulation software generates output files that are not UTF-8 encoded. As a result, there could be content in these output files that OASIS cannot read by default; this causes errors when an optimization is run. To check the format of your simulation’s output file, please use a document editor such as Notepad++ as it will show you the encoding of a file. [OASIS Variables](#)
 - c. If suggestions a) and b) do not suffice, please acquire the OASIS log files and reach out to support@empowerops.com.
2. Saved project file cannot be opened in OASIS.
 - a. The first step is to ensure you are using the correct version of OASIS to open the saved project file. The structure of how project files are saved may potentially be altered between releases of OASIS and as such, in most cases saved project files from older versions of OASIS may not work with newer versions of OASIS. If you have access to an older version of OASIS and need to retrieve data from these saved project files, please contact support@empowerops.com.
 - b. It is always recommended to save a copy of an opyl configuration file prior to working on any optimizations. Doing this allows problem configurations to be saved for use on varying versions of OASIS, as our opyl configuration feature is

designed to allow users to port existing problems between different versions of OASIS. Note that actual optimization data is not stored in these opyl files.

3. Error balloons are showing up in OASIS.
 - a. Warnings and Errors in OASIS both show up in the same location at the bottom right corner. If a user runs into some sort of error while using OASIS the user will either see a red or yellow balloon. Yellow balloons are only associated with warnings and only indicates issues that should not hinder a user's current workflow in OASIS. If you encounter a yellow balloon message please notify us at support@empowerops.com with the steps you did prior to the balloon showing up.
 - b. On the other hand, red balloons are only associated with errors. The specific error may range from not having a proper simulation integration output file when an optimization is running to a button in OASIS that doesn't work. Because these errors can range from something very miniscule to issues that are larger than initially perceived, we recommend that the first course of action would be to go double check the state of the problems. For example, if a user runs into a red error balloon when the user clicks 'optimize' after setting up a problem definition we would advise the user to double check the problem to make sure there's no issues with it. Since red error balloons, unlike yellow warning balloons may potentially be workflow blocking for any user we advise users to reach out to support@empowerops.com with the following information:
 - i. The version of OASIS.
 - ii. Your operating system.
 - iii. The summary of the problem you are trying to solve and at which step the error occur.
 - iv. The relevant files (if they are not private files) included with the email.

15.2 Logging

Most actions done in OASIS are logged. That is, if a user runs into some error then a stack trace (a report that provides information about a program's subroutines) is saved in the log files. This information is extremely handy for debugging errors. We understand that some users may have

experience reading stack traces and that other users may not have this experience. In both cases, we recommend any relevant stack traces to be sent to support@empowerops.com so that our support team can get a clear picture of the issue. Sending the relevant stack traces to our support team will also help speed up any debugging process. Logs, by default, are stored in the: 'C:\Users\<<your username>\AppData\Local\Temp\Empower Operations\<<OASIS version>\logs' directory. They are stored as .log files.



<input type="checkbox"/> Name	Date modified	Type	Size
 OASIS_0_0.log	2017-05-07 3:35 PM	Text Document	283 KB
 test-logs-0.log	2017-05-05 3:00 PM	Text Document	3,522 KB

Figure 84 - Example of Log Files