# TRACE-SIMEXP v1.0 - Scripting Utility for Computer Experiment of TRACE

# STARS Memorandum

Date: 23.06.2016

From: D. Wicaksono

Phone: +41 56 310 2759

Loc.: OHSA/D08

Email: damar.wicaksono@psi.ch

To:

1. STARS / O. Zerkak
2. STARS / Y. Aounallah
3. ERP / G. Perret

cc:

1. STARS / STARS\_TRACE
2. STARS / M. Krack
3. STARS / H. Ferroukhi

# Abstract

A computer experiment is a multiple model runs using different values of the model parameters. Its design, in particular the selection of the design points at which the model will be evaluated, well as its analysis, in particular the analysis of the output variation in relation to the inputs variation, are useful for sensitivity and uncertainty analyses of the model subjected to the experimentation.

An important prerequisite of carrying out such experiment is the availability of a supporting tool able to handle the related logistical aspects. a Python3-based scripting utility has been developed to assist in carrying such experiments for TRACE code. The scope of the utility is ranging from the pre-processing the TRACE input deck amenable for batch parallel execution to the post-treatment of the resulting binary xtv file amenable to subsequent sensitivity and uncertainty analyses. This memorandum describes the development of the tool, including the description on its usage, implementations, and assumptions.

# Introduction and Scope of the Utility

In the context of PSI contribution to the OECD/NEA PREMIUM Benchmark Phase IV, there was a need to be able to execute numerous TRACE runs of a given model with different values of the input parameters in a systematic manner. By taking the developments from the design and analysis of computer experiment, a reasonable number of inputs combinations (directly translated to the number of code runs) can be judiciously selected and the resulting outputs can then be analyzed further. These were done either for the purpose of quantifying the model prediction uncertainty (*uncertainty propagation*) or of understanding better the input/output relationship of the complex model (*sensitivity analysis*). This document details the implementation of a Python3-based utility to assist in carrying out computer experiment for TRACE model.

The top-level flowchart of the utility is shown in Figure 1. The process is done in three sequential steps with clearly defined inputs and outputs as well as a set of required command line arguments. The flowchart also explicitly states the scope of the utility development that begins with a set of input files and ends with a set of csv files. The most important input file for a computer experiment campaign is the design matrix which contains normalized input parameters values at which TRACE will be run. The design matrix is produced by generic numerical routine and is outside the scope of trace-simexp. The set of csv files, each of which contains TRACE graphic variables extracted from the produced binary xtv files is the ultimate output of the utility. The csv files are ready to be further post-processed for sensitivity/uncertainty analysis. The tools required to carry out such an analysis is also outside the scope of trace-simexp.

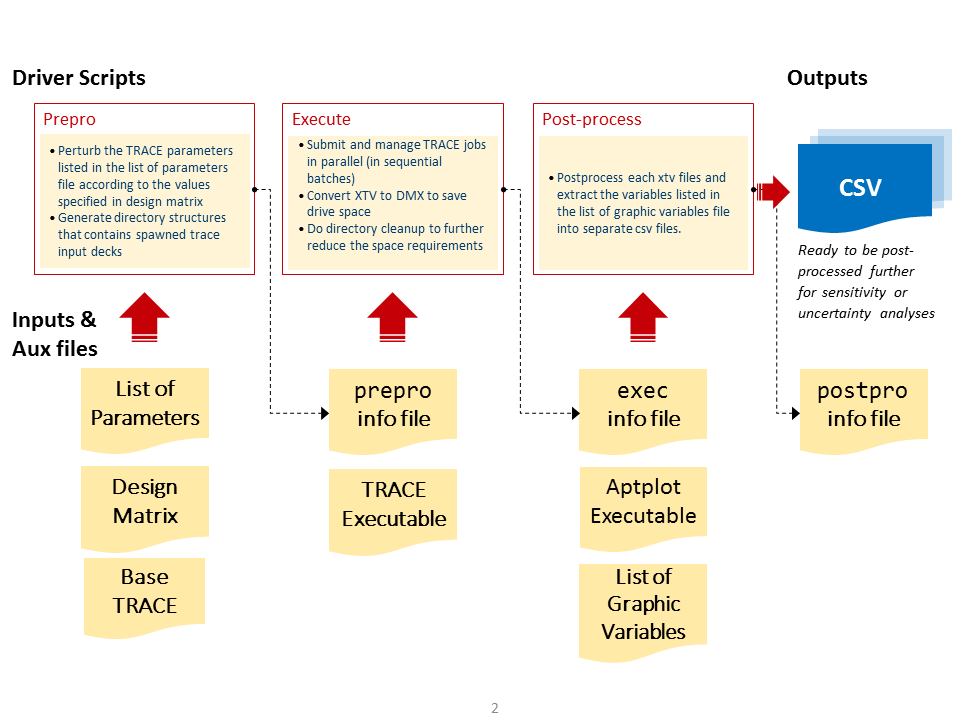


Figure 1: Generic flowchart of trace-simexp including the required input files

In the subsequent sections, the usage information as well as the notes on the implementation will be explained in more detail.

# List of Features (v1.0)

The current version and all its features were consolidated from the developments made for the PSI contributions to the PREMIUM Phase-IV benchmark (the application of uncertainty propagation), the NUTHOS-11 conference (the application of the Morris method), and the NURETH-16 conference (the application of the Sobol’ method). All the aforementioned applications were related to TRACE reflood model on the basis of FEBA separate effect facility for reflood experiment.

The features of this release are:

1. Complete separation of the process in 3 different steps: prepro, exec, and postpro. At each step an auxiliary file (so called info file) is produced and used at the subsequent step (except the postpro info file) as well as for documentation and diagnostic purposes.
2. Three modes of parameter perturbation are supported: additive, multiplicative, and substitutive.
3. Four categories of TRACE variables in the input deck that can be perturbed: spacer grid, material properties, *sensitivity coefficients*, and components.
4. Specifically for the TRACE components, five different components are supported: PIPE, VESSEL, POWER, FILL, BREAK.
5. Specification of the computer experiment by the users is done through a set of input files (list of parameters file, design matrix file, and list of graphic variables file) and a set of command line arguments given at each step.
6. Iso-probabilistic transformation of the normalized design matrix is available for uniform, discrete uniform, log-uniform, and normal distributions.

# Usage

## Step 1: Preprocessing

In the preprocessing step, the base TRACE input deck is modified by changing the parameter values of the parameters listed in the list of parameter files according to the values listed in the design matrix file. A set of new perturbed TRACE input decks will be created into separate directories. In subsequent execute step, these directories will serve as the run directories. The preprocessing step driver script can be invoked in the terminal using the following command:

python prepro.py {-as, -ns, -nr} <argument to select samples to create> \  
 -b <the base run directory name> \  
 -tracin <the base TRACE input deck> \  
 -dm <the design matrix> \  
 -parlist <the list of parameters file> \  
 -info <The short description of the campaign> \  
 -ow <flag to overwrite existing directory structure>

Brief explanation on this parameter can be shown using the following command:

python prepro.py --help

The table below lists all the arguments in detail.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Short Name | Long Name | Type | Required | Description | Default |
| 1 | -as | –all\_sample | flag | Yes, iff -nr or -ns not supplied | Preprocess all samples in design matrix | False |
| 2 | -ns | –num\_samples | integer(s) | Yes, iff -as or -nr not supllied | Preprocess the selected samples | None |
| 3 | -nr | –num\_range | 2 integers | Yes, iff -as or -ns not supplied | Preprocess the range of samples, inclusive | None |
| 4 | -b | –base\_name | string | No | The base run directory name | ./simulation |
| 5 | -tracin | –base\_tracin | string | Yes | The base TRACE input deck, path+filename | None |
| 6 | -dm | –design\_matrix | string | Yes | The design matrix, path+filename | None |
| 7 | -parlist | –params\_list | string | Yes | The list of parameters file, path+filenam | None |
| 8 | -info | –info | string | No | Short message of the experiment | None |
| 9 | -ow | –overwrite | flag | No | Flag to overwrite existing directory structure | False |

The directories created is nested in the following form:

.  
|  
+---<the base run directory name>  
| +---<tracin>  
| +---<name based on dm and parlist>  
| +---<tracin-run\_1>  
| tracin-run\_1.inp  
| +---<tracin-run\_2>  
| tracin-run\_2.inp  
| +---<tracin-run\_3>  
| tracin-run\_3.inp  
|  
...

In addition to the creation of the run directory structure and perturbed TRACE input deck, the script execution will also produce an info file (from here on in will be called *prepro info file*). The info file is produced by default with the following naming convention:

prepro-<tracin name>-<parlist name>-<dm name>-<sample\_start>\_<sample\_end>.info

The file is used to document the command line arguments specified when the script was called. It will also be used in the subsequent step.

**Example**

For example, upon executing the following command:

python prepro.py -as \  
 -b ./simulation \  
 -tracin ./simulation/base/febaTrans214.inp \  
 -dm ./simulation/dmfiles/optLHS\_110\_2.csv \  
 -parlist ./simulation/paramfiles/febaVars2Params.inp \  
 -info "FEBA Test No. 214, 110 Samples, 2 Parameters"

A set of directory will be created

.  
|  
+---simulation  
| +---febaTrans214  
| +---febaVars7Params-optLHS\_110\_2  
| +---febaTrans214-run\_1  
| febaTrans214-run\_1.inp  
| +---febaTrans214-run\_2  
| febaTrans214-run\_2.inp  
| +---febaTrans214-run\_2  
| febaTrans214-run\_3.inp  
...  
| +---febaTrans214-run\_110  
| febaTrans214-run\_110.inp

Based on the command above, the prepro info file will be created with the following name:

prepro-febaTrans214-febaVars2Params-optLHS\_110\_2-1\_110.info

The file has the following (abridged) contents:

TRACE Simulation Experiment - Date: 2016-03-27 00:21:07.196979  
FEBA Test No. 214, 110 Samples, 2 Parameters  
\*\*\*Preprocessing Phase Info\*\*\*  
Base Name -> simulation  
Base Directory Name -> ./simulation  
Base Case Name -> febaTrans214  
Base Case File -> ./simulation/base/febaTrans214.inp  
List of Parameters Name -> febaVars2Params  
List of Parameters File -> ./simulation/paramfiles/febaVars2Params.inp  
Design Matrix Name -> optLHS\_110\_2  
Design Matrix File -> ./simulation/dmfiles/optLHS\_110\_2.csv  
Overwrite Directory -> 0  
Samples to Run ->  
1 2 3 4 5 6 7 8 9 10  
 ...  
101 102 103 104 105 106 107 108 109 110  
\*\*\* 1\*\*\*  
Sensitivity Coefficient with ID \*1039\* is specified  
Parameter type: scalar  
Parameter perturbation mode: 3 (multiplicative)  
Parameter distribution: logunif  
1st distribution parameter: 0.250  
2nd distribution parameter: 4.000  
\*\*\* 2\*\*\*  
Sensitivity Coefficient with ID \*1011\* is specified  
Parameter type: scalar  
Parameter perturbation mode: 3 (multiplicative)  
Parameter distribution: logunif  
1st distribution parameter: 0.500  
2nd distribution parameter: 2.000

## Step 2: Execute

In the execute step, all the input decks that were created in the preprocessing step are executed sequentially in batch. This means that the script will traverse the run directories created before and execute the input deck inside sequentially. The size of a batch is controlled by the number of processors supplied by the user through the command line argument.

Simultaneous execution of multiple TRACE simulation often requires large amount of disk space even for a single case. To save disk space, the utility takes two measures. First, the binary xtv file is not written directly in the running directory during the execution. Instead a soft link is created inside the running directory, linked to the actual xtv file written in a *scratch* directory. This approach was adopted to limit the disk space usage in a STARS project working directory (or *the activity folder*) that is a backup volume and limited to 200 [GB] currently. The so-called *scratch* directory usually resides in a non-backup volume. Second, after each execution, the resulting xtv file will be directly converted to the more space efficient *dmx* format. This is done by using xtv2dmx utility. As such, the path to the scratch directory as well as the path to the executable for xtv2dmx utility are needed to be supplied during the call.

The execute step driver script can be invoked in the terminal using the following command:

python execute-py -prepro <the preprocessing step info file> \  
 -nprocs <the number of available processors> \  
 -{ns, nr, as} <selection of samples to be executed> \  
 -scratch <the scratch directory> \  
 -trace <the trace executable> \  
 -xtv2dmx <the xtv2dmx executable>

Brief explanation on the required arguments can be printed on the screen using the following command:

python execute.py --help

The table below lists all the arguments used in the execute.py driver script.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Short Name | Long Name | Type | Required | Description | Default |
| 1 | -prepro | –prepro\_file | string | Yes | The path to the preprocess info file | None |
| 2 | -nprocs | –num\_processors | integer | No | The number of processors to use for execution | 1 |
| 3 | -as | –all\_sample | flag | Yes, iff -nr or -ns not supplied | Preprocess all samples in design matrix | False |
| 4 | -ns | –num\_samples | integer(s) | Yes, iff -as or -nr not supplied | Preprocess the selected samples | None |
| 5 | -nr | –num\_range | 2 integers | Yes, iff -as or -ns not supplied | Preprocess the range of samples, inclusive | None |
| 6 | -scratch | –scratch\_directory | string | Yes | The path of the scratch directory | None |
| 7 | -trace | –trace\_executable | string | Yes | The path to the TRACE executable | None |
| 8 | -xtv2dmx | –xtv2dmx\_executable | string | Yes | The path to the XTV2DM executable | None |

The script execution will also produce an info file (from here on in will be called *exec info file*). The info file is produced by default with the following naming convention:

exec-<tracin name>-<parlist name>-<dm name>-<sample\_start>\_<sample\_end>.info

The file is used to document the command line arguments specified when the script was called, to log the process run for diagnostic purpose, as well as be used in the subsequent (postpro) step. See below for the example of the contents.

**Example**

Following the previous example, executing the following command will execute all of the TRACE input decks created in the previous step using 5 processors (or, parallel jobs with multiple batches each of size 5).

python execute.py -prepro prepro-febaTrans214-febaVars2Params-optLHS\_110\_2-1\_110.info \  
 -as \  
 -scratch /afs/psi.ch/group/lrs/scratch/grp.lrs.scr001.nb/wicaksono\_d/ \  
 -trace trace\_v5.0p3.uq\_extended \  
 -xtv2dmx xtv2dmx\_v6.5.2\_inst01.sh \  
 -nprocs 5 >& 214\_1060\_7.log &

**Remarks**: The utility was so far tested in the lclrs machines. To keep the kerberos token active for a long session, it is advised to use the k5run -B command and put the job in the background with the following command instead:

k5run -B python execute.py -prepro prepro-febaTrans214-febaVars2Params-optLHS\_110\_2-1\_110.info \  
 -as \  
 -scratch /afs/psi.ch/group/lrs/scratch/grp.lrs.scr001.nb/wicaksono\_d/ \  
 -trace trace\_v5.0p3.uq\_extended \  
 -xtv2dmx xtv2dmx\_v6.5.2\_inst01.sh \  
 -nprocs 5 >& 214\_1060\_7.log &

Based on the command above, the prepro info file will be created with the following name:

exec-febaTrans214-febaVars2Params-optLHS\_110\_2-1\_110.info

The file has the following (abridged) contents:

TRACE Simulation Experiment - Date: 2016-03-27 00:26:06.547934  
\*\*\*Execute Phase Info\*\*\*  
prepro.info Filename -> prepro-febaTrans214-febaVars2Params-optLHS\_110\_2-1\_110.info  
TRACE Executable -> trace\_v5.0p3.uq\_extended   
XTV2DMX Executable -> xtv2dmx\_v6.5.2\_inst01.sh   
Scratch Directory Name -> /afs/psi.ch/group/lrs/scratch/grp.lrs.scr001.nb/wicaksono\_d  
Number of Processors -> 5 (lclrs73)  
Samples to Run ->   
 1 2 3 4 5 6 7 8 9 10  
 11 12 13 14 15 16 17 18 19 20  
...  
101 102 103 104 105 106 107 108 109 110  
\*\*\* Batch Execution - 1 \*\*\*  
Execution Successfull: trace\_v5.0p3.uq\_extended -p febaTrans214-run\_1  
Execution Successfull: trace\_v5.0p3.uq\_extended -p febaTrans214-run\_3  
Execution Successfull: trace\_v5.0p3.uq\_extended -p febaTrans214-run\_5  
Execution Successfull: trace\_v5.0p3.uq\_extended -p febaTrans214-run\_2  
Execution Successfull: trace\_v5.0p3.uq\_extended -p febaTrans214-run\_4  
Execution Successfull: xtv2dmx\_v6.5.2\_inst01.sh -r febaTrans214-run\_1.xtv -d febaTrans214-run\_1.dmx  
Execution Successfull: xtv2dmx\_v6.5.2\_inst01.sh -r febaTrans214-run\_3.xtv -d febaTrans214-run\_3.dmx  
Execution Successfull: xtv2dmx\_v6.5.2\_inst01.sh -r febaTrans214-run\_5.xtv -d febaTrans214-run\_5.dmx  
Execution Successfull: xtv2dmx\_v6.5.2\_inst01.sh -r febaTrans214-run\_2.xtv -d febaTrans214-run\_2.dmx  
Execution Successfull: xtv2dmx\_v6.5.2\_inst01.sh -r febaTrans214-run\_4.xtv -d febaTrans214-run\_4.dmx  
...  
\*\*\* Batch Execution - 22 \*\*\*  
Execution Successfull: trace\_v5.0p3.uq\_extended -p febaTrans214-run\_106  
Execution Successfull: trace\_v5.0p3.uq\_extended -p febaTrans214-run\_108  
Execution Successfull: trace\_v5.0p3.uq\_extended -p febaTrans214-run\_110  
Execution Successfull: trace\_v5.0p3.uq\_extended -p febaTrans214-run\_107  
Execution Successfull: trace\_v5.0p3.uq\_extended -p febaTrans214-run\_109  
Execution Successfull: xtv2dmx\_v6.5.2\_inst01.sh -r febaTrans214-run\_106.xtv -d febaTrans214-run\_106.dmx  
Execution Successfull: xtv2dmx\_v6.5.2\_inst01.sh -r febaTrans214-run\_108.xtv -d febaTrans214-run\_108.dmx  
Execution Successfull: xtv2dmx\_v6.5.2\_inst01.sh -r febaTrans214-run\_110.xtv -d febaTrans214-run\_110.dmx  
Execution Successfull: xtv2dmx\_v6.5.2\_inst01.sh -r febaTrans214-run\_107.xtv -d febaTrans214-run\_107.dmx  
Execution Successfull: xtv2dmx\_v6.5.2\_inst01.sh -r febaTrans214-run\_109.xtv -d febaTrans214-run\_109.dmx

## Step 3: Postprocessing

After all the requested TRACE input decks (or samples) have been executed, the resulting xtv files can be post-processed to extract the relevant variables and put them into separate csv files, placed inside the respective running directory. The csv files, being text files, can be easily processed further with other tool for various purposes. Similar to the execute step before, the utility will traversed each of the executed running directory and process the xtv file inside using the aptplot program to extract the requested variables. The script also supports batch parallel execution.

The postprocessing step driver script can be invoked in the terminal using the following command:

python postpro.py -exec <the execute phase info file> \  
 -aptplot <the aptplot executable> \  
 -nprocs <the number of available processors> \  
 -vars <the list of TRACE graphic variables file>

Brief explanation on the required arguments can be printed on the screen using the following command:

python postpro-py --help

The table below lists all the required arguments in detail.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Short Name | Long Name | Type | Required | Description | Default |
| 1 | -exec | –exec\_info | string | Yes | the path to the execute info file | None |
| 2 | -aptplot | –aptplot\_executable | string | Yes | the path to aptplot executable | None |
| 3 | -nprocs | –num\_processors | integer | No | The number of processors to use for postprocessing | 1 |
| 4 | -vars | –trace\_variables | string | Yes | Preprocess the selected samples | None |

**Remarks 1**: When running the script interactively under Windows which is connected to an lclrs machine, the aptplot program requires an X Server running in the background (e.g., Xming).

**Remarks 2**: Make sure there is enough disk space in the running directory as the csv files being produced. Depending on how many variables are being extracted, the files (being a text file) can take considerable amount of disk space.

In addition to the postprocessing of the xtv files, the execution of postpro script will also produced an info file (hereinafter *postpro info file*). The info file is produced by default with the following naming convention:

postpro-<tracin name>-<parlist name>-<dm name>-<sample\_start>-\_<sample\_end>-<vars name>.info

The file is used to document the command line arguments specified when the script was called as well as to log all the shell commands run during the execution. See below for example of the contents.

**Example**

Following the previous example, executing the following command will postprocessed all of the TRACE xtv files produced in the previous step using 5 processors (or, parallel jobs with multiple batches each of size 5) to extract TRACE graphic variables listed in the ./simulation/listVars.apt file.

python postpro.py -exec exec-febaTrans214-febaVars2Params-optLHS\_110\_2-1\_110.info \  
 -aptplot aptplot\_v6.5.2\_inst01.sh \  
 -nprocs 5 \  
 -vars ./simulation/listVars.apt

**Remarks**: Similar to the execute phase, if the postprocess step is expected to take a long time it is advised that the job is sent to the background with k5run -B utility as given in the previous example.

Based on the command above, the prepro info file will be created with the following name:

postpro-febaTrans214-febaVars2Params-optLHS\_110\_2-1\_110-listVars.info

The file has the following (abridged) contents:

TRACE Simulation Experiment - Date: 2016-03-24 17:34:10.713350  
\*\*\*Post-process Phase Info\*\*\*  
prepro.info Filename -> prepro-febaTrans214-febaVars2Params-optLHS\_110\_2-1\_110.info  
exec.info Filename -> exec-febaTrans214-febaVars2Params-optLHS\_110\_2-1\_110.info  
APTPlot Executable -> aptplot\_v6.5.2\_inst01.sh   
Number of Processors (Host) -> 5 (lclrs73)  
List of XTV Variables Files -> ./simulation/listVars.apt  
List of XTV Variables ->   
 rdzNperm-20A01 rdzNperm-20A02 rdzNperm-20A03  
 rdzNperm-20A04 rdzNperm-20A05 rdzNperm-20A06  
 rdzNperm-20A07 rdzNperm-20A08 rdzNperm-20A09  
 ...  
 rftn-20A141R29 rftn-20A142R29   
Samples to Post-processed ->   
 1 2 3 4 5 6 7 8 9 10  
 11 12 13 14 15 16 17 18 19 20  
 21 22 23 24 25 26 27 28 29 30  
 31 32 33 34 35 36 37 38 39 40  
 41 42 43 44 45 46 47 48 49 50  
 51 52 53 54 55 56 57 58 59 60  
 61 62 63 64 65 66 67 68 69 70  
 71 72 73 74 75 76 77 78 79 80  
 81 82 83 84 85 86 87 88 89 90  
 91 92 93 94 95 96 97 98 99 100  
101 102 103 104 105   
\*\*\* Batch Execution - 1 \*\*\*  
Execution Successful: aptplot\_v6.5.2\_inst01.sh -batch febaTrans214-run\_1-listVars.apt -nowin  
Execution Successful: aptplot\_v6.5.2\_inst01.sh -batch febaTrans214-run\_3-listVars.apt -nowin  
Execution Successful: aptplot\_v6.5.2\_inst01.sh -batch febaTrans214-run\_5-listVars.apt -nowin  
Execution Successful: aptplot\_v6.5.2\_inst01.sh -batch febaTrans214-run\_2-listVars.apt -nowin  
Execution Successful: aptplot\_v6.5.2\_inst01.sh -batch febaTrans214-run\_4-listVars.apt -nowin  
...

## List of parameters file (params\_list file)

The list of parameters file contains the specification of the selected TRACE parameters to be perturbed during the experiment. The file also contains the specification of the perturbation factor. The user specified all of the required information in a text file and for each line contains the information listed in the table.

|  |  |  |
| --- | --- | --- |
| No. | Name | Description |
| 1 | enum | enumeration of specified parameter in the list |
| 2 | data\_type | type of the parameters |
| 3 | var\_num | parameter ID number, typically a unique TRACE input deck ID |
| 4 | var\_name | parameter name |
| 5 | var\_type | parameter type |
| 6 | var\_mode | mode of perturbation |
| 7 | var\_card | **card**, where the specific perturbed parameter is located |
| 8 | var\_word | **word**, where the specific perturbed parameter is located |
| 9 | var\_dist | Distribution of the perturbation factor (as random variable) |
| 10 | var\_par1 | the 1st parameter of the distribution |
| 11 | var\_par2 | the 2nd parameter of the distribution |
| 12 | str\_fmt | string formatting of the parameter within the trace input deck |

### data\_type

The available parameters which can be accessed during the computer experiment are classified into several categories or, referring to the table above, data\_type. The currently supported *data\_types* are:

1. Spacer grid model specification (keyword spacer)
2. Material properties (keyword matprop)
3. TRACE *sensitivity coefficients* (keyword senscoefs)
4. TRACE components, specifically for pipe, vessel, power, fill, and break

The variables var\_num, var\_type, var\_card, var\_word differ from type to type and will be explained in their corresponding subsection.

### var\_mode

A perturbation factor will be assigned for each of the specified parameter. There are three modes of perturbation according to the table.

|  |  |
| --- | --- |
| var\_mode | Description |
| 1 | substitutive, the sampled factor is directly substitutes the nominal parameter value |
| 2 | additive, the sampled factor is added to the nominal parameter value |
| 3 | multiplicative, the sampled factor is multiplied by the nominal parameter value |

### var\_dist, var\_par1, var\_par2

In trace-simexp, the perturbation factor associated with each specified model parameter is modeled as random variable. The variables var\_dist, var\_par1, and var\_par2 are required to fully specify the probability density (or *mass*, if discrete) function of the random variable. These specifications are used to transform the normalized value given in the design matrix file into the actual perturbed TRACE model parameter value.

The table below describes the currently supported univariate distribution and the meaning of the variables. The type of the variable is written in the bracket.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Name | var\_dist | var\_par1 (type) | var\_par2 (type) |
| 1 | uniform | uniform | minimum value (float) | maximum value (float) |
| 2 | discrete uniform | discunif | list of choices with equal probability (list) | N/A |
| 3 | log-uniform | logunif | minimum value (float) | maximum value (float) |
| 4 | normal (gaussian) | normal | mean (float) | variance (float) |

### str\_fmt

The variable str\_fmt in params\_list file specified the string formatting of the perturbed parameter when it is writen in the TRACE input deck as a string. The formatting follow the conventional printf format string such as 14.4e, 14d, 14.4e, etc.

**Remarks**: In the current version, the users have to check by themselves the accordance between perturbed parameter and its string format in the input deck. The complete specification on the string formatting for TRACE input deck (in FORTRAN) can be found in the User’s Manual Vol. 1.

### Comment symbol

The utility supports in-line commenting through the use of hash character.

**Example**

The following lines are ignored by the parsing utility and was written down as a guidance for user

# 0 1 2 3 4 5 6 7 8 9 10 11  
# enum data\_type var\_num var\_name var\_type var\_mode var\_card var\_word var\_dist var\_par1 var\_par2 str\_fmt

### Spacer Grid Model Parameters

### Material Properties

### TRACE *Sensitivity Coefficient*

The term *sensitivity coefficient* was introduced in the special delivery of trace\_v5.0p3 and now it becomes a standard feature of the new release of TRACE (trace\_v5.0p4). This coefficient, in principle, is simply a perturbation factor applied to TRACE closure laws parameters (e.g., heat transfer coefficient or interfacial drag) and made available to the user via the input deck. As such, the term is a misnomer and it is always written in this document in italic.

An example of how sensitivity coefficient is defined in the input deck is given below,

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* Model flags \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*  
.....  
\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* Sensitivity Coefficients \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*  
\* Spacer Grid Pressure Loss Coefficient Multiplier  
\*id mode value  
1033 3 1.0  
\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* component-number data \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*  
......

The \*sensitivity coefficients inside the input deck requires three variables: id, a unique integer number identifying the coefficient (see table); mode, the mode of perturbation (see table); value, the actual value of perturbation factor.

The table below gives all the required information, to specify the *sensitivity coefficient* in the list of parameters file. Not the if a variable is not used it has to be specify with - (i.e., dash symbol) inside the list file.

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Description | Value |
| 1 | enum | enumeration in the list | integer |
| 2 | data\_type | type of parameters | senscoef |
| 3 | var\_num | unique integer ID for the *sensitivity coefficient* | (see table) |
| 4 | var\_name | **not used** | - |
| 5 | var\_type | type of variable | scalar |
| 6 | var\_mode | mode of perturbation | (see table) |
| 7 | var\_card | **not used** | - |
| 8 | var\_word | **not used** | - |
| 9 | var\_dist | distribution of the perturbation factor | (see table) |
| 10 | var\_par1 | the 1st parameter of the distribution | (see table) |
| 11 | var\_par2 | the 2nd parameter of the distribution | (see table) |
| 12 | str\_fmt | string formatting of the parameter | 14.4f |

**Example**

An example of how a *sensitivity coefficient* is specified inside the list of parameters file is shown below

# 0 1 2 3 4 5 6 7 8 9 10 11  
# enum data\_type var\_num var\_name var\_type var\_mode var\_card var\_word var\_dist var\_par1 var\_par2 str\_fmt  
 16 senscoef 1035 - scalar 3 - - logunif 0.5 2.0 14.4f

The example above showed the perturbed parameter no. 16 of type *sensitivity coefficient* applied to the input as a multiplication factor with log-uniform distribution between 0.5 to 2.0.

### TRACE Component Parameters

## Design matrix file (design\_matrix file)

The design matrix (or *the design of experiment*) file is a text file that contains the sampled and normalized (values between 0 - 1) input parameters values. The matrix is of dimension *N x K*, where *N* corresponds to the number of rows (i.e., the number of samples) and *K* corresponds to the number of columns (i.e., the number of parameters/dimensions). The values of each rows can be separated by comma, tab, space, or semicolon.

The values inside the file can be generated by various different procedures such as the simple random sampling (SRS), Latin Hypercube Sampling (LHS), or quasi-random sequence (Sobol’ Sequence, Halton Sequence, Hammersley set, etc) each with its own statistical property. The procedure to generate the values is outside the scope of trace-simexp utility. Example of tools to generate such values is Simlab [1](https://ec.europa.eu/jrc/en/samo/simlab), DiceDesign (an R package [2](https://cran.r-project.org/web/packages/DiceDesign/index.html)), or OpenTurns (a Python module [3](http://www.openturns.org/)).

**Example**

The excerpt below is taken from the first 7 lines of a design matrix file with 5 parameters separated by comma.

3.154313e-02,5.621832e-01,1.046621e-01,6.978293e-02,1.599945e-01  
1.475773e-01,1.229935e-01,7.299532e-01,8.951946e-01,4.153665e-01  
2.229898e-01,9.981891e-01,3.783502e-02,1.558353e-01,5.350678e-01  
9.846914e-01,7.849868e-02,4.686996e-01,2.895825e-01,6.436200e-01  
9.250452e-01,6.194709e-01,6.873275e-01,5.125879e-01,8.598690e-01  
1.974221e-01,4.532642e-01,4.914747e-01,6.819097e-01,8.206347e-02  
2.832927e-01,8.773572e-01,3.679600e-01,5.648471e-01,2.747036e-01  
...

## List of graphic variable file (trace\_variables file)

The list of (TRACE) graphic variable file is a simple text file that contains in each row, a single TRACE graphic variable name to be extracted from the xtv output file. The extracted values will be in time-series over all the transient. The complete documentation of the graphic variables can be found in Chapter 3 of the TRACE User’s Manual Volume 1.

**Example**

An example of the content of the file is the following

rftn-20A11R29  
rftn-20A12R29  
rftn-20A13R29  
...

The three variables above correspond to the evolution of temperature for HtStr No. 20 at axial levels 11, 12, 13 and radial node 29, respectively.

## TRACE and aptplot (trace\_executable and aptplot\_executable)

## CSV Output File

# Implementation

## Assumptions

## Known Limitations

## Distribution

# Examples of Use Cases

# References