Direction de l'Energie Nucléaire Centre de Cadarache Département d'Etudes des Réacteurs Service de Simulation en THermohydraulique	CEA/DEN/GRE/DER/SSTH/LMDL DO 34 03/04/09 09PPQM000038

Note technique DEN

Document Méthodologique

Organisation de la base de données des cas test de validation de Trio_U. Cas test de validation automatisés.

Trio_U validation test cases data base organisation. Automatic Validation or verification test cases.

DEN/CAD/DER/SSTH/LMDL/NT/2009-108/A

S. Vandroux, V. Barthel, G. Fauchet

Commissariat à l'Energie Atomique
Centre : GRENOBLE
Adresse : 17 avenue des Martyrs 38054 Grenoble Cedex
Tél. : 04 38 78 40 53 Fax : 04 38 78 94 53 Courriel : bernard.faydide@cea.fr
Établissement public à caractère industriel et commercial
RCS Paris B 775 685 019

NIVEAU DE CONFIDENTIALITE				
DO	DR	CCEA	CD	SD
X				

PARTENAIRES/CLIENTS	ACCORD	TYPE D'ACTION
CEA		

REFERENCES INTERNES CEA			
DIRECTION D'OBJECTIFS	DOMAINE	PROJET	EOTP
DSOE	**Simulation**	**Sithy**	**A-Sithy-03-03-01**
JALON	INTITULE DU JALON	DELAI CONTRACTUEL DE CONFIDENTIALITE	CAHIERS DE LABORATOIRE

SUIVI DES VERSIONS			
INDICE	DATE	NATURE DE L'EVOLUTION	PAGES, CHAPITRES
A	03/04/09	Emission initiale	**31, 4**

	NOM	FONCTION	VISA	DATE
REDACTEUR	S. Vandroux	Ingénieur		20/4/09
VERIFICATEUR	O. Cioni	Ingénieur		21/4/09
CONTROLE QSE	P. Boudier	Ingénieur qualité		02/06/09
APPROBATEUR	F. Ducros	Chef de laboratoire		04/06/09
EMETTEUR	B. Faydide	Chef du SSTH		05/06/2009

MOTS CLEFS

Validation, organisation, cas test automatisé, base de données, Trio_U

RESUME / CONCLUSIONS

Cette note présente l'organisation de le base de données de validation de Trio_U.

Cette base de donnée est composée de deux types de cas test : les anciens cas 'faits à la main' avant la version 1.5.5, et les nouveaux cas automatisés.

La manière de générer ces cas tests automatisés est tout d'abord présentée. On passe ensuite en revue les différentes procédures automatiques mises en place qui sont lancées après la mise en gestion de configuration d'un nouveau cas test.

Le classement des cas tests dans la base de données de validation est ensuite présenté. Deux types de classifications existent : une liste exhaustive des cas tests rangés suivant une certaine logique où chaque cas n'apparait qu'une fois, ainsi qu'une matrice multi-niveaux qui donne la liste des cas tests correspondant à une certaine liste de paramètres.

La matrice multi-niveaux permet de visualiser l'état de validation du code : ce qui est validé et ce qui a encore besoin de l'être.

The way to generate the new automatic test cases is first explained. A review of the automatic launching procedures after the versioning of a new test case is then described.

The classification of the test cases in the validation data base is then outlined. Two classifications exist: either an exhaustive list of the validation test cases calculated by Trio_U classified with certain logic where each test case appears only once, or a multi-level matrix giving lists of test cases fitting with a given list of parameters.

The multi-level matrix enables to see the validation state of the code: what is validated and what still needs to be validated.

DIFFUSION (COURRIEL) :

DEN/DSOE :	C. CHAULIAC
DEN/DER/DIR :	G. MIGNOT
DEN/DER/SSTH :	B. FAYDIDE, D. TENCHINE, D.JAMET
DEN/DER/SSTH/LDAS :	M. FARVACQUE
DEN/DER/SSTH/LDAL :	F. DUCROS, , V. BARTHEL, U. BIEDER, O. CIONI G. FAUCHET, T. FORTIN, C. FOURNIER, E. GARRÉ, O. LEBAIGUE B. MATTHIEU, F. PERDU P. QUEMERE, N. TAUVERON, S. VANDROUX
DEN/DER/SSTH/LDLD :	P. EMONOT, P. BOUDIER
DEN/DTN/SE2T/LITA :	M. BERTHOUX
DEN/DER/SESI :	J.Y. MALO
DEN/DER/SESI/LCFR :	N. DE VICTOR
DEN/DER/SESI/LCSI :	D. PLANCQ
EQUIPE DE MAINTENANCE CSSI :	P. LEDAC, C. MALOD, M. ELMO

Table of contents

1. INTRODUCTION

The objective of this document is to present the new organisation of the Trio_U validation data base.

For the time being, this data base is composed of two different kinds of test cases:

- The classical test cases which were run up to the version 1.5.4 of Trio_U. The calculation, the post processing and the corresponding validation report were 'hand-made' once. In order to update them, all the work has to be done again.
- The new test cases, run from version 1.5.5, where a procedure enables to run all the calculations corresponding to this test case automatically. The post processing and the validation report are also done automatically. These test cases are very easily updated for a new version of the code. Besides, with this method, it is easy to compare the calculation results of the code with the results of the different versions.

The first part of this report describes the procedure used to generate these automatic test cases, how they are classified, and how a new test case is added to the data base.

The second part of the report describes the organisation of the validation data base in different physical problems. It will particularly show how the different types of thermohydraulic cases are organised in a test case list and in a multi level validation matrix.

2. TEST CASES

This section is devoted to the test cases which are generated automatically. These test cases are generated on Linux PCs or on UNIX machines. The way to generate a new test case is described. The different procedures following the generation of a test case are also explained.

2.1 General principle of how to generate an automatic test case

The objective of the automatic procedure is:

- To run all the calculations corresponding to a given test case.
- To post process the results and generate the needed tables and figures.
- To generate a 'report' file in PDF format which contains all the necessary information for someone willing to have a close look at a given test case.
- To compare easily calculations performed with different versions of the code

In order to generate an automatic test case, the TRIO_U user will first create a directory devoted to this test case in his work domain.

In this directory, he will create a directory '**src**' (for 'source'). This directory will contain all the files necessary to create the automatic test case, and nothing more. Among the necessary files, there will be a file with a .**prm** extension. Once this directory filled properly as will be explained below, the user will launch the following command to obtain the report:

$TRIO_U_ROOT/Validation/Outils/Genere_courbe/Run src/*file*.prm

2.1.1 Files necessary in the src directory

In order to achieve this, the directory src will have to contain several files. In the simplest case, the necessary files will be a data file and the ***file*.prm** which contains the parameters of the automatic report generation.
The other possible files are:

- Files necessary to launch a calculation: other data files, meshing files…
- Files which are not generated automatically but which are in the final report (for example, pictures of test descriptions, experimental data used for comparison…)
- All the optional procedures (prepare, prerun, postrun) mentioned in the next section (2.1.2).

2.1.2 Execution of the Run command line

This command will first copy the directory src and its containing files in another directory named '**build**'. In this directory build,

- the calculations are prepared (generation of directories, data files…) in accordance with the content of a file named '**prepare**'
- The calculations listed in the ***file*.prm** are then launched one after the other.
- For each calculation:
  - o If necessary, prior to the calculation, some treatment can be needed. This is done according to a '**prerun**' file
  - o The calculation is performed
  - o After each calculation, if necessary, some post processing is done using a '**postrun**' file
- Thanks to the ***file*.prm** the different figures or tables which need to be seen are created, if necessary, by using a '**pre-requis**' command, and a final report is generated with the text contained in ***file*.prm** and with the figures and tables.

2.1.3 Quick description of Run command line options

The Run command **$TRIO_U_ROOT/Validation/Outils/Genere_courbe/Run src/*file*.prm**
is described briefly in the file
$TRIO_U_ROOT/Validation/Outils/Genere_courbe/Doc_utilisation

It has several interesting options which enable to generate the final report with more efficiency when a new test case is generated.

- ***-o toto.pdf*** enables to choose the name of the final report. By default, it will be 'rapport.pdf'
- ***-not_run*** enables to launch the generation of the pdf file without launching all the calculations. It is very useful to produce the report file
- ***-post_run*** enables to launch the generation of the pdf file without launching all the calculations but launches all the post run automatic commands. It is very usefull to correct graphs or tables.
- ***-v 1*** generates the file test_lu.prm in the build directory. This file contains interesting information, such as the number of the different figures produced in the report file
- ***-d text*** only the text in the final report is modified accordingly to the changes of the file.prm
- ***-d nn*** only the figure number nn is generated. The final report is not modified.
- ***--compare*** Described in section 2.1.4.
- ***-archive*** Described in section 2.5.2

In case of new developments of the **Run** tool, the user who has subscribed on the following web site:
http://saxifrage:3500/wws/info/trio_u_post_traitement
will be informed.
On this site, the user can also post his questions, and both the question and the answer will be communicated to all the subscribers.

2.1.4 Tips and recommendation to users

During the finalization of his test case, the user will be tempted to modify some files in the 'build' directory before running the 'Run' command.
This must be avoided, because when the 'Run' command is launched, the files present in the 'src' directory will be copied in the 'build' directory. The modified files in the 'build' directory will be overwritten; the modifications will be lost and not taken into account.

When launching the 'Run' command with different versions of the code, the 'build' directory will be overwritten. If the user wants to save the results of the previous version of the code, he must rename the directory 'build' (in build_version1 for exemple).

Once this is done, using the –compare option of the 'Run' command, the user can then compare the results obtained with his two (or more) versions with the command:
$TRIO_U_ROOT/Validation/Outils/Genere_courbe/Run src/*file*.prm --compare=../build_version1/ --compare=../build_version2/
This will compare the requested (in file.prm) meshes, tables and figures of current version, version 1 and version 2 and generate a report file for comparison.
N.B. All the items are compared, except those for which the 'Origine xxx' indication (in **file.prm**) is specified, where xxx is different from 'Trio_U'.

2.2 Automatic test case generation step by step

As was seen in section 2.1.2, the Run command line has several steps. This section explains more concretely how to generate an automatic test case.

2.2.1 How to use a prepare file

This 'prepare' file is a shell script. It is an optional file, but in most test cases, it is necessary to use one. Most of the time, the validation cases need more than one data file because several options are tested. The **prepare** file can thus be used :
-to create sub directories in the build directory
-to copy the appropriate files in these directories
-to generate the different data files necessary for the calculation starting from a single data file

Here is an example of a 'prepare' file from the test case 'T_paroi':

```
#!/bin/sh

# Creation of several directories
mkdir -p Nu_impose U_impose Symet

# Appropriate files copied in the directories
for dir in Nu_impose U_impose Symet
do
        cd $dir;   cp ../first.awk ../post_run ../T_paroi.data  . ; cd ..
Done
```

*# **Modification of the initial data file in several directories***
cd Symet
sed "s/paroi_fixe/symetrie/" T_paroi.data > modifie.data ; mv modifie.data T_paroi.data
cd ../U_impose
*(sed "s/Loi_Paroi_Nu_Impose { nusselt 20.*Re^(0.)*Pr^(0.) diam_hydr champ_uniforme 1 2.14e-3 }/Loi_standard_hydr_scalaire/" T_paroi.data | sed "s/{ u_tau Champ_uniforme 1 0. }/{ u_tau Champ_uniforme 1 0.2 }/" > modifie.data ; mv modifie.data T_paroi.data)*
cd ..

2.2.2 How to use a prerun or a post_run file

This 'prerun' or 'postrun' files are also shell scripts. They are optional files. In most cases, only a post_run file is needed. They are useful if some treatment is needed prior to the calculations or if some post processing is needed after the calculations. Depending on the test case, their content can vary from a few lines to lots of shell, awk or python command lines.

Here is an example of post_run file for the expansion_3D_VDF_VEF test case:

#!/bin/sh
*# **Extraction of a curve using a pressure probe***
extrait_coupe 3d_vdf_rect SONDE_PRESSION
*# **Execute a file containing awk command lines***
sh ../first.awk 3d_vdf_rect
*# **Fill a data file which will be used by the file.prm to draw a figure or fill a table***
more delta_p.dat >> deltap_courbe.dat

An example of a 'first.awk' file extracted from the test case 'T_paroi' is given in Annex 1.

2.2.3 How to use a prm file

An example of a prm file is given in Annex 2 (T_paroi.prm), and the corresponding report file is given in Annex 3.
The commands contained in the prm file will briefly be commented here with the help of the example T_paroi.prm. However, the user will find an exhaustive description of all the tools used in the prm files in the following file:
$TRIO_U_ROOT/Validation/Outils/Genere_courbe/doc/manuel.xhtml
In the bottom of this file, several examples are available, giving both the prm file and the resulting pdf report.

A first chapter (Parametres: see Annex 2 and Annex 3) is filled with several important parameters of the calculation.

Different chapters can then be included. In each chapter, the user can include:

- **Tables** defined in the prm file. An interesting option ('Tableau_performance') enables to issue directly the computer performances with one command line.
- **Figures** either coming from existing image files or from gnuplot files created by the prm file
- **Visualisations** (visu) which can be obtained directly from the lata files generated by the Trio_U calculations.

Latex format can be integrated easily in the description parts of the chapters.

For example:

Example for a Table for test case T_paroi

Tableau { Titre "Second calculation results"
Description "Here, the wall friction velocity is imposed. Thus, the wall temperature gradient is calculated with this u* value. The theorical value of ΔTw is given thanks to the Kader law. We compare it with the Trio_U results, resumed in the table below."
Description " "
nb_colonnes 6
label Tmean Outlet | Tfluid | Twall (face) | ΔTw | Twall "equiv" | ΔTw "equiv"
Ligne { Legende "Theorical value"
fichier ./U_impose/theoric.dat }
Ligne { Legende "Trio_U paroi_fixe"
fichier ./U_impose/temp.dat
origine trio_u } }

Will be transformed into the following section with table:

5.2 Second calculation results

Here, the wall friction velocity is imposed. Thus, the wall temperature gradient is calculated with this u* value. The theorical value of ΔTw is given thanks to the Kader law. We compare it with the Trio_U results, resumed in the table below.

	Tmean Outlet	Tfluid	Twall (face)	ΔTw	Twall equiv	ΔTw equiv
Theorical value	0.1333	-	-	5.6791	-	5.6791
Trio_U paroi_fixe	0.1333	0.4629	0.5377	0.0747	6.1421	5.6791

figure 1 Example for a table

Example for an existing figure for test case T_paroi:

Figure { Titre "New Boundary conditions"
Width 8cm
Image ./T_paroi_geo2.png }

Will be transformed, thanks to the image T_paroi_geo2.png into the following figure:

5.1 New Boundary conditions

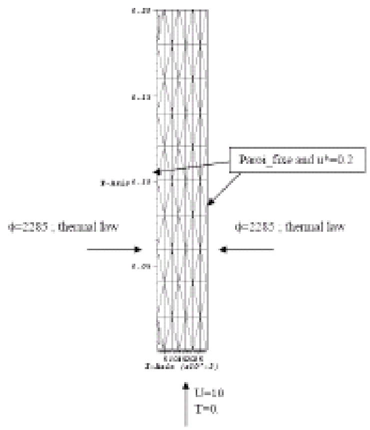


figure 2 Example for an existing figure

Example for a figure created by the prm file for test case expansion_2D_axi:

Figure { Titre "Axial pressure distribution for inlet velocity =1m/s at t=8s"
Dimension 2
Description "The graph below shows the pressure distribution along the axis of the pipe"
Description "The value of P1 is taken for an axial distance equal to zero."
LabelX "Axial distance (in m)"
LabelY "Pressure (in kPa)"
InclureDescCourbes 0
Courbe { Legende "-axial pressure"
Origine "Trio_U"
Version "1.5.5"
Segment ./2d_vdf_scz_1/2d_vdf_scz sonde_pression
Style linespoints
colonnes ($1) ($2) } }

Will be transformed in the following figure:

3.4 Axial pressure distribution for inlet velocity =1m/s at t=8s

The graph below shows the pressure distribution along the axis of the pipe
The value of P1 is taken for an axial distance equal to zero.

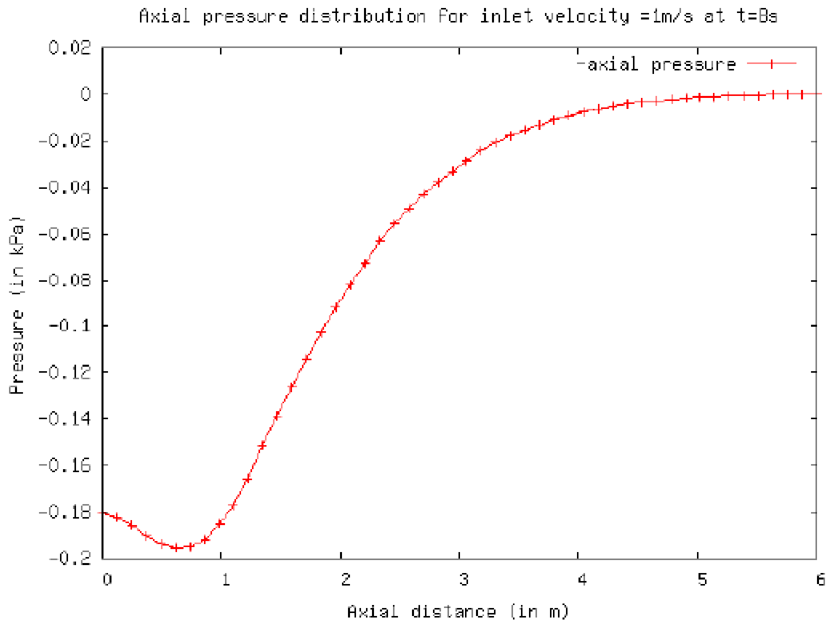


figure 3 Example for a figure created by the prm file

Example for a Visualisation for test case expansion_2D_axi

Visu { titre "Velocity distribution at t=8s"
Width 9 cm
vector ./2d_vdf_scz_1/2d_vdf_scz.lata dom VITESSE ELEM }

3.1 Velocity distribution at t=8s

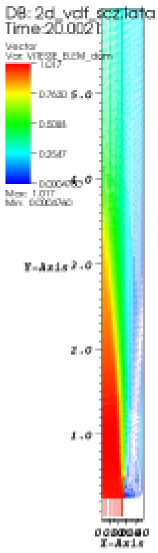


figure 4 Example for a visualisation

2.2.4 Tips and recommendations to users

Some simple rules have to be respected when generating a new test case.

- All the generated report files in pdf must be written in English. It is however admitted to have comments in French in the scripts.
- In the introduction chapter, the user must specify his name. The name of a 'permanent' member of the Trio_U team who has followed the work must be added if the user only works temporarly in the Trio_U team.
- If the user wants to write a section in his report that will have neither table nor picture, he must use the command 'tableau'
- The user should do the necessary in order to have as little data sets as possible in his src directory. The objective is of course to have just one data set, though it is sometimes not possible or difficult. The reason for this is that in case of a new version of the code, where the syntax of the data set has changed, it will be much quicker to modify just one data set instead of more.
- Once the new test case is generated and ready to be versioned, there should be only one prm file left in the src directory, in order to avoid confusion.
- The src directory should only contain the strict necessary in order to create the test case.

2.3 Organisation of the test cases

Once they exist, these test cases generated on Linux PCs or on UNIX machines must be stored in order to be taken into account in the next release version of the code.

All the existing test cases available for a version of Trio_U can be found in:
$TRIO_U_ROOT/Validation/Rapport_automatiques

This directory is divided into several directories:

- Validant/Fini: storage of the finished validating test cases.
- Validant/pas_fini: storage of the validating test cases which still need some work to be done (text in the report file, translation in English, test description…)
- Verification_codage: storage of the test cases which are designed to verify some parts of the computation
- Problemes_en_cours: storage of the test cases for which problems were found which are not solved yet, but which need to be solved in the future, or least, for which a track has to be kept.

2.4 Versioning of a test case

Once the user is ready to submit his test case to the validation team, he must use the clearcase versioning tool in order to store his test case in one of the above storage directory.

Once located in the proper directory (normally in Validant/Fini), a special script is launched by typing:

- $TRIO_U_ROOT/Validation/Outils/Genere_courbe/scripts/new_test_case_versioning test_name
  (where test_name is the name of the test case)

- **CHECKIN_ALL**

then type "**ok**" or any other comment.

Once warned, the maintenance team will deliver the src directory of the new test case with the next version of the code in the $TRIO_U_ROOT/Validation/Rapport_automatiques directory.

The test case can than be submitted to the validation team, which will then approve or suggest potential corrections.

2.5 Following prodedures

Once the test case has been versioned in the code, several procedures described below are applied.

2.5.1 Reference test case

From each test case versioned in the Rapports_automatiques directory, non regression test cases are generated automatically. They consist in the data files used in the test cases but with only a few time steps.
It is launched with each new version of the code.
If differences are observed with the previous version, the maintenance team checks if there is an evolution of the syntax in the new version.

If it is the case, the necessary syntax is modified in the validation test case. The non regression test case is generated automatically. The non regression test case test case is launched again and the results are compared to those of the previous version.

If differences are detected for a test case, it will appear on a list of test cases which are to be launched again (automatically) and checked by the validation team. The PDF report can be compared to the previous one by using the --compare option of the Run command. Depending on the result, the validation team will decide if the new developments can be integrated in the version or not.
If these differences disable the release of the new version of the code, the origins of these differences should then be spotted and corrected before the new release.

2.5.2 Automatic launch of the test cases

A Linux PC called timber launches automatically all the existing versioned automatic test cases. For each version of the code, a ValidationXXX (where XXX is the code version) directory is generated.

The final report of the test cases and (using the –archive option of the Run command) an archive file containing the files necessary for the report file are created. They are stored in the ValidationXXX/archives directory. This directory is available to every person of the Trio_U team working on a Linux PC and connected to the intranet.

The new versioned test cases are spotted by an automatic procedure launched every day. Once spotted, the test_case directory is launched automatically as soon as possible, according to priority rules defined in this launching procedure.

2.5.3 General report

Once all the test cases calculated, the PDF files can very easily be merged in a global PDF file in order to create a global validation test case report, with an interactive table of contents.

3. VALIDATION MATRIX (CARACAS)

Once created and versioned with the procedures seen in the previous chapter, the new test case is classified in a validation data base.

Explaining the organisation of this validation data base is the aim of this chapter.

As already mentioned, for the time being, this data base is composed of two different kinds of test cases:
- The classical test cases which were run from version 1.4 up to version 1.5.4 of Trio_U. with a ‘hand-made’ validation report
- The new test cases, run from version 1.5.5, using the procedures described in chapter 2.

This data base is available on the DENShare zone of the server CARACAS (S:) at the following address: S:\300_PROJETS\315_TRIO_U\315b_TRIO-U_Valid\validation.
The necessary authorization is required to access this address for the people who do not belong to the Trio_U team.

At this address, the user will have access to several “doc” or “pdf” files and several directories.

Each directory contains a validation test case. If it is a hand made test case, the directory contains different files necessary for the test case (data file, report file…). If it is an “automatic” test case, the directory contains the report file in PDF format and the archive file (see 2.5.2).
The validation team is in charge of updating the directories with a ‘reference’ report file corresponding to the expected results of the validation test case. The validation team is also in charge to check the results of this test case on further versions of the code.

In order to access the different validation tests more easily, the test cases are gathered in two word files with two different logics.
These word files:
- Validation_list_of_cases.doc and
- Validation_matrix_two_levels.doc

are described below.

3.1 List of test cases

The file Validation_list_of_cases.doc contains the exhaustive list of the validation test cases calculated by Trio_U.

The test cases are classified depending on different aims or problems:

- Numerics: gives access to the validation test cases designed to test numerical options of the code, for example the convection schemes.
- Laminar flow hydraulic: give access to hydraulic laminar flow problems
- Laminar flow thermohydraulic: for thermohydraulic laminar flow problems.
- RANS modelling hydraulic: for mixing length or k epsilon turbulent modelling of hydraulic flow problems.
- RANS modelling thermohydraulic: for mixing length or k epsilon turbulent modelling of thermohydraulic flow problems
- LES modelling hydraulic: for Large Eddy Simulation in hydraulic problems
- LES modelling thermohydraulic: for Large Eddy Simulation in thermohydraulic problems
- Front tracking: for problems using the front tracking option

- Radiation: for problems taking radiation into account.
- Porous media: for porous media problems.

Each problem appears only once in one of these designations. For each designation, a table gives the list of the available test cases.

An example is given in the following table for the 'RANS modelling thermohydraulic' designation

Title	Modelled problem	Tested functionnality	Code version	Discr.	Author	Tested against
Forced_Convection_EF_stab	Turbulent flow in a heated circular pipe	3D k-ε model + Pr_t for Na	1.5 5	VEFP1B	V. Barthel	Analytical Solution
Convection_kEps_QC	Turbulent flow in a heated square cavity	2D k-ε model QC and non QC model	1.4.8	VDF VEFP1B	G. Garnier (V. Barthel)	EDF Experiment
Channel_T1_T2_VEF	Turbulent flow in plane 3D channel	3D k eps QC and incompressible model Wall heat exchane	1.5.5	VEFP1B	C.Fournier (C.Malod)	Nu correlations
stratified_flow	Turbulent mixing layers	K eps, boussinesq, buyancy	1.5.5	VDF VEFP1B	V.Barthel	Experiment

Table 1 : List of the available test cases for the RANS modelling thermohydraulic

As can be seen, the first column of the table contains the name of the test case. This name is an hypertext link. A mouse click on this link opens the directory corresponding to the test case.
The links have three different colours. The brown colour corresponds to the validating automatic test. The green colour corresponds to the automatic test cases which are validating, but which still need some improvements (corresponds to Validant/pas_fini in section 2.3). The blue colour corresponds to the validating test cases which are not yet generated automatically.

The second column contains a quick description of the modelled problem.

The third column contains a quick description of the tested functionalities.

The fourth column specifies the code version. For the automatic test cases, the code version is also mentioned in the report file.

The fifth column specifies the author of the test case. If it has been done by someone staying only temporarily, the name of the permanent person 'in charge' should be specified.

The last column enables to see with what kind of data the test cases is validated (experiment, theory, correlations, analytic calculations, other CFD calculations…)

The validation team is in charge of updating this file.

The classification of the test cases in the file Validation_list_of_cases.doc enables to have a quick overview of the existing test cases. As already mentioned, each test case appears only once.

3.2 Multi level matrix

In order to have a more precise overview of the validation test cases, another classification is also made. This new classification uses matrixes with different levels. It can be found in the file Validation_matrix_two_levels.doc.

The objective of this classification is to define a set of parameters (type of problem, type of discretization, type of turbulence model…). For this set of parameters it must be possible to visualize both:

- The existing test cases
- The validation gap, in order to be able to define the needed test cases to improve the validation of the code.

Considering the significant number of possible parameters which can be investigated in the code, it is difficult to define an exhaustive and permanent set of parameters.

The ideal solution would be to generate automatic multi-level validation matrixes for given set of parameters. This work should be performed later. In the meantime, a set of important parameters has been chosen and has led to the construction of a two level matrixes system.
The first level starts from the different type of problems.
The second level starts from the discretization type, the time scheme and the turbulence model.

Attempts were made to add a third level containing all the other parameters (geometry, boundary conditions, source terms, physical properties, convection scheme, diffusion scheme …). As these matrixes are updated by hand by the validation team however, the necessary work to create and update them was judged to be too tedious.

In this multi level matrix system, each test case can appear several times, depending on the investigated parameters.

3.2.1 First level matrix

In this matrix, the test cases are classified by the type of problem.
The main different possible problems are the following:

- Hydraulic
- Hydraulic concentration
- Thermohydraulic
- Thermohydraulic QC (quasi compressible)
- Thermohydraulic concentration
- Conduction
- Problems with passive scalars
- Porous problems
- Front Tracking

Among these problems, most of the test cases concern hydraulic, thermohydraulic or thermohydraulic QC problems.
The first level matrix is thus reduced to the below table. It will be possible to complete this table if new test cases are developed for other problems.
In this table, the grey boxes correspond to impossible combinations. The light blue boxes corresponding to the diagonal of the table correspond to 'simple' problems (among the above list). The light orange boxes correspond to the coupled problems.

In the light blue box corresponding to the simple problems, an hypertext link gives the total number of test cases treating the given problem. It leads to a second level matrix described in the next section. The light orange boxes correspond to the coupled problems. They contain a list of hypertext links. Each link gives the name of a test case and leads to the directory containing the test case. In the future, when the test list is more significant, an hypertext link will give the number of test cases leading to a second level matrix, as it is done for the simple problems.

The colour legend of the links is the same as in the previous section:

- Brown automatic test validating tests.
- Green: automatic test validating tests which need improvements
- Blue: test cases not generated automatically.

	Hydraulic	Thermohydraulic	Thermohydraulic QC	Conduction	Front tracking
Hydraulic	22				
Thermohydraulic		17			
Thermohydraulic QC			7		
Conduction		2D_coupling_Pb Nusselt_Correlation_Coupling_Pb Conduction_couple		0	
Front tracking	Bulle_3D_VDF_FT Bulle_3D_VEF_FT Ftd_gravite				1

Table 2 : First level matrix

3.2.2 Second level matrix

The second level matrix only concerns the light blue boxes of the first level matrix corresponding to the simple problems. A mouse click on the hypertext link leads to a second level matrix described here.

For illustration, an example of a second level matrix for the incompressible thermohydraulic problem is given in the table below.

	Explicit (or RK2 or RK3)		CN itérative		Piso
	VDF	VEF	VDF	VEF	VEF
laminar	Laminar_Flow_Vertical_Plate Oscillating_Flow Nusselt_Correlation_2D_Channel (LES dégradé)	Comp_conv Convection_rotating_table Convection_Vahl_Davis Upwind_Muscl_and_Limiters_VEF Laminar_Flow_Vertical_Plate		Comp_conv Convection_rotating_table Convection_Vahl_Davis	

		Oscillating_Flow Diagonal_Cube_Transport Nusselt_Correlation_2D_Channel (LES dégradé) T_paroi (LES dégradé)			
Mixing length					
K eps	stratified_flow	stratified_flow Forced_Convection_EF_stab Channel_T1_T2_VEF			
LES	(Channel_T1_T2_VDF_LES_ReT180) (Channel_T0_Q_VDF_LES_ReT406) (Convection_mixing_length)	(Convection_mixing_length)		Channel_T1_T2_VEF_LES_ReT180	

Table 3 : Example of second level matrix

The boxes contain a list of hypertext links. Each link gives the name of a test case and leads to the directory containing the test case.

The colour legend of the links is the same as in the previous section:

- Brown automatic test validating tests.
- Green: automatic test validating tests which need improvements
- Blue: test cases not generated automatically.

With this example, it can be seen that there exist no mixing length test cases. No test case exists either for the Piso time scheme.

Some test cases such as ‘Comp_conv’ or ‘stratified_flow’ appear in different boxes

4. SUMMARY AND CONCLUSION

This report presents the new organization of the Trio_U validation data base. This data base is composed of two different kinds of test cases: ‘old hand made’ test cases for the versions before 1.5.5 of Trio_U, and new automatic test cases.

A user can generate an automatic validation test case. For this, he has to work on a Linux PC or a UNIX machine. To create automatic test case, he uses the tools which are released with the Trio_U version. Once the test case is finished, it must be submitted to the Trio_U validation team. Once it is approved, the user must do the versioning of his test case. This enables the other users to access to the src directory of his test case. It is released with each new version of the code. After the release of each new version of Trio_U, the test case is launched automatically. The resulting files are available to every person of the Trio_U team working on a Linux PC and connected to the intranet. The test case is also used to generate a non regression test case, which will partly condition the release of a new version of the code.

A global document containing the reports of all the automatic test cases performed with a code version can be issued

The full validation data base is available by the Trio_U team and the people having the required permission on the DENShare zone of the server CARACAS. This zone is updated by the validation team. For each test case, it contains both the report and either an archive file or several usefull files. It also contains two different classifications of the test cases:

- An exhaustive list of the validation test cases calculated by Trio_U classified with a certain logic. Each test case appears only once.
- A multi-level matrix giving lists of test cases fitting with a given list of parameters.

The multi-level matrix enables to see the validation state of the code: what is validated and what still needs to be validated.

In order to improve the validation state of the code, the new test cases will be developed in the automatic form. In parallel, all the old test cases will have to be transformed into automatic ones.

In order to spot even better the validation state of the code, future developments will enable to generate automatic multi-level validation matrixes for a set of parameters chosen by the user.

Reference

Description of the Run command:
$TRIO_U_ROOT/Validation/Outils/Genere_courbe/Doc_utilisation

Information on the new developments concerning the Run command.
http://saxifrage:3500/wws/info/trio_u_post_traitement

Description of all the tools used in the prm files :
$TRIO_U_ROOT/Validation/Outils/Genere_courbe/doc/manuel.xhtml

Available existing test cases:
$TRIO_U_ROOT/Validation/Rapport_automatiques

Data base of the available test cases
DENShare zone of the server CARACAS (S:):
S:\300_PROJETS\315_TRIO_U\315b_TRIO-U_Valid\validation

Annex 1: Example of first.awk file

```
#!/bin/sh
# Post processing for the 2D channel: data written temp.dat
tps_init_moyen=0
LC_NUMERIC=C

# Necessary data read in data file
mu=`grep "mu" $1.data| awk '{print $4}' | head -1`
rho=`grep "rho" $1.data| awk '{print $4}' | head -1`
lambda=`grep "lambda" $1.data| awk '{print $4}' | head -1`
Cp=`grep "Cp" $1.data| awk '{print $4}' | head -1`

echo $mu > ../mu.dat
echo $rho > ../rho.dat
echo $Cp > ../Cp.dat
echo $lambda > ../lambda.dat

Prandtl=`awk "BEGIN { print $mu*$Cp/$lambda }" `

# Calculation of theoretical  Tmoyen and DeltaTw
X=`grep -i "longueur" $1.data| awk '{print $2}' | head -1`
Y=`grep -i "longueur" $1.data| awk '{print $3}' | head -1`
NX=`grep -i "Nombre_de_Noeuds" $1.data| awk '{print $2}' | head -1`
D=`awk "BEGIN { print $X/($NX-1)/2.} "`

phi=`grep "flux_impose" $1.data| awk '{print $5}' | head -1`

V=`grep "vitesse_imposee" $1.data| awk '{print $6}' | head -1`
Tmoyen=`awk "BEGIN {print 2*$phi*$Y/($rho*$Cp*$V*$X)}" `
tail -3 $1_pbf_Nusselt.face > fin_nusselt

# Calculation of theoretical temperature gradient
# If imposed Nu, direct calculation, else, use of Kader
logi=`grep -i "Loi_Paroi_Nu_Impose" $1.data | head -1`
if [ logi == "" ]
then
  # reading of u* which should be different frome 0, Calculation of  y+andt T+ with Kader
  uetoi=`grep "u_tau" $1.data| awk '{print $7}' | head -1`
  yplus=`awk "BEGIN {print $D*$uetoi*$rho/$mu}" `
  beta=`awk "BEGIN {print (3.85*$Prandtl^(1/3)-1.3)^2+2.12*log($Prandtl)}" `
  gamma=`awk "BEGIN {print 0.01*($Prandtl*$yplus)^4/(1+5*$yplus*$Prandtl^3)}" `
  tplus=`awk "BEGIN {print $Prandtl*$yplus*exp(-$gamma)+(2.12*log(1+$yplus)+$beta)*exp(-1/$gamma)}" `
  # DeltaTw=tplus*t_tau is deduced
  t_tau=`awk "BEGIN {print $phi/($uetoi*$Cp*$rho)}" `
  DeltaTw=`awk "BEGIN {print $t_tau*$tplus}" `
else
  Nu=`more fin_nusselt| awk '{print $7}' | head -1`
  DeltaTw=`awk "BEGIN {print $phi*$D/($lambda*$Nu)}" `
fi
awk -v Tmoy=$Tmoyen -v dT=$DeltaTw 'BEGIN {printf ("%.4f %s %s %.4f %s %.4f\n",Tmoy,"-","-",dT,"-",dT)}'  > theoric.dat
```

#reading of Tmoyen Trio_U
TSmoyen=`tail -1 Tmoyen_sortie | awk '{print $2}' | head -1 `

#reading of Tfluide et Tparoi Trio_U
Tfluide=`more fin_nusselt| awk '{print $11}' | head -1`
Tparoi=`more fin_nusselt| awk '{print $13}' | head -1`
Tparoi_eq=`more fin_nusselt| awk '{print $15}' | head -1`

#Calculation of thes deltaT Trio_U at the wall
deltaT=`awk "BEGIN {print $Tparoi-$Tfluide}" `
deltaT_eq=`awk "BEGIN {print $Tparoi_eq-$Tfluide}" `

#filling of the result file temp.dat
awk -v Tmoy=$TSmoyen -v Tf=$Tfluide -v Tp=$Tparoi -v Tp_eq=$Tparoi_eq -v dT=$deltaT -v dT_eq=$deltaT_eq 'BEGIN {printf ("%.4f %.4f %.4f %.4f %.4f %.4f\n",Tmoy,Tf,Tp,dT,Tp_eq,dT_eq)}' > temp.dat

exit

Annex 2: example of a prm file: T_paroi.prm

```
Parametres {
        Titre "Wall temperature verification in VEF discretisation with Neumann conditions"
        Description "Calculations with imposed u*, and Nusselt number imposed or free"
        Auteur "V.B."
        CasTest "Nu_impose" "T_paroi.data" data file printed in report file
        CasTest "Symet" "T_paroi.data"
        CasTest "U_impose" "T_paroi.data"
        VersionTrio_U "1.5.6_beta"
}

Chapitre {
        Titre "Model description"

Figure  {
        Titre "Geometry, Mesh and Boundary Conditions"
        Description "Dimensions : h=200 mm, L=30 mm "
        Description "7 x 10 rectangular mesh cut with '"trianguler' option"
        Description "Velocity inlet = 10 m/s ; Temperature inlet = 0°C"
        Description "Lateral Heat flux = 2285 W/m2"
        Description "Outlet, pressure = 0"
     Width 8cm
     Image ./T_paroi_geo1.png
        }

Tableau {
        Titre "Physical properties"
     Description "Fluid domain: Helium"
     Description "No Gravity effect"
        nb_colonnes 1
                Label Valeur
          Ligne {
                Legende "$\rho$ en kg/m$³$"
               fichier ./rho.dat
          }
          Ligne {
                Legende "$\mu$ en N/m$²$/s"
               fichier ./mu.dat
          }
          Ligne {
                Legende "$\lambda$ en W/m/K"
               fichier ./lambda.dat
          }
          Ligne {
                Legende "Cp en J/kg/K"
               fichier ./Cp.dat
          }
        }
}

Chapitre {
        Titre "Numerical Results"
Tableau {
```

Titre "Analytical solutions for two calculation conditions"
Description "The mean outlet temperature is deduced from global energy balance :"
Description "\latex_($\rho\:Cp\:U\Delta Tes = \Phi,\: so \:\Delta Tes = 0.1\:K$\latex_) where ΔTes is the temperature difference between inlet and outlet."
Description ""
Description "The Nusselt number in Trio_U is a local heat exchange coefficient. Knowing this Nu number, we can deduce the theorical temperature gradient at the wall, i.e. the difference between wall and first fluid point temperatures :"
Description "\latex_($\Delta Tw = \Phi\:d(\lambda\:Nu)$\latex_)where Δ Tw is this wall temperature difference, it will be taken at the channel outlet."
Description "In our case, the distance between these two points is d = 2.14e$^-$^{3}m"
Description "When we impose a thermal wall law (Kader), the local gradient, and so the Nusselt number, is calculated with this law"
}
}

Chapitre {
Titre "First calculation : u=0 and Nu=20"*
Tableau {
Titre "Comparisons of mean and wall temperature value at the outlet"
Description "As defined before, the theorical ΔTw can be known. We show in the following table, the different values calculated by Trio_U and compare with the analytical solution."
Description "The first ΔTw value corresponds to the difference between the first fluid temperature value and the face temperature at the wall. Thus it is disturbed by the convection term discretization."
Description "The second ΔTw "equiv" value corresponds to the difference between the first fluid temperature value and the equivalent wall temperature calculated with the local Nusselt number in Trio_U. "
Description " "
Description "The same calculation has be done with two different boundary conditions at the wall : 'symetrie' or 'paroi_fixe' to see the effect of non-tangential velocities at the edge"
Description " "
nb_colonnes 6
label Tmean Outlet | Tfluid | Twall (face) | ΔTw | Twall "equiv" | ΔTw "equiv"
Ligne {
Legende "Theorical value"
fichier ./Nu_impose/theoric.dat
}
Ligne {
Legende "Trio_U paroi_fixe"
fichier ./Nu_impose/temp.dat
origine trio_u
}
Ligne {
Legende "Trio_U symet"
fichier ./Symet/temp.dat
origine trio_u
}
}
}

Chapitre {
Titre "Second calculation : u=0.2 and thermal law function"*
Figure {
Titre "New Boundary conditions"
Width 8cm

```
        Image ./T_paroi_geo2.png
          }

Tableau {
        Titre "Second calculation results"
        Description "Here, the wall friction velocity is imposed. Thus, the wall temperature gradient is calculated with this u* value. The theorical value of $\Delta$Tw is given thanks to the Kader law. We compare it with the Trio_U results, resumed in the table below."
        Description " "
        nb_colonnes 6
          label Tmean Outlet | Tfluid | Twall (face) | $\Delta$Tw | Twall "equiv" | $\Delta$Tw "equiv"
          Ligne {
               Legende "Theorical value"
               fichier ./U_impose/theoric.dat
          }
          Ligne {
               Legende "Trio_U paroi_fixe"
               fichier ./U_impose/temp.dat
                        origine trio_u
          }
     }
Tableau {
        Titre "Conclusions"
        Description "We can note than the temperature gradient calculated with the Trio_U face temperature (*) is very far from the analytical one. This is due to the convection terms in the VEF discretization method."
        Description "Because of this convection effect, the modification of the local Nusselt number as no visible effect on the wall temperature."
        Description "When we take the wall "equivalent" temperature calculated with the thermal wall law, or the local Nusselt number (*), the $\Delta$Tw_eq is perfectly correct."
        Description "We can assure that the total wall heat is brougth to the fluid since the mean temperature at the channel outlet is correct."
        Description " "
        Description "(*): this temperature values can be found in the "T_paroi_pbf_Nusselt.face" output file."
     }

tableau_performance {
        Titre "Computing performance"
     }
}
```

Annex 3: example of report file: T_paroi

Wall temperature verification in VEF discretisation with Neumann conditions

1 Introduction

Validation made by : V.B..
Report generated 19/01/2009.

1.1 Description

Calculations with imposed u*, and Nusselt number imposed or free

1.2 Parameters Trio_U

- Version Trio_U : 1.5.6_beta
- Version Trio_U from out: /work/triou/Validation/Trio_U_mpi_opt (1.5.6_beta)

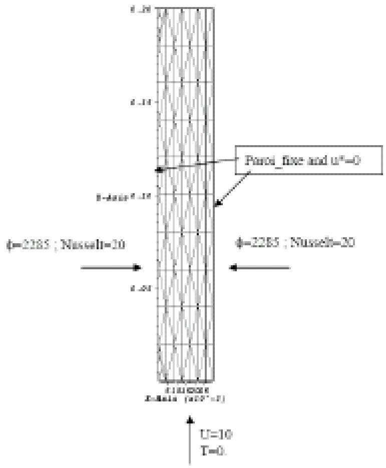
1.3 Test cases

- Nu_impose/T_paroi.data : *impression du jeu de données en fin de fichier*
- Symet/T_paroi.data :
- U_impose/T_paroi.data :

2 Model description

2.1 Geometry, Mesh and Boundary Conditions

Dimensions : h=200 mm, L=30 mm
7 x 10 rectangular mesh cut with 'trianguler' option
Velocity inlet = 10 m/s ; Temperature inlet = 0 C
Lateral Heat flux = 2285 W/m2
Outlet, pressure = 0



2.2 Physical properties

Fluid domain: Helium
No Gravity effect

	Valeur
ρ en kg/m^3	4.4
μ en N/m^2/s	4.4e-05
λ en W/m/K	0.34
Cp en J/kg/K	5193.0

3 Numerical Results

3.1 Analytical solutions for two calculation conditions

The mean outlet temperature is deduced from global energy balance :
$\rho\, Cp\, U \Delta Tes = \Phi$, so $\Delta Tes = 0.1\ K$ where ΔTes is the temperature difference between inlet and outlet.

The Nusselt number in Trio_U is a local heat exchange coefficient. Knowing this Nu number, we can deduce the theorical temperature gradient at the wall, i.e. the difference between wall and first fluid point temperatures :
$\Delta Tw = \Phi\, d(\lambda\, Nu)$where Δ Tw is this wall temperature difference, it will be taken at the channel outlet. In our case, the distance between these two points is d = 2.14e^{-3}m
When we impose a thermal wall law (Kader), the local gradient, and so the Nusselt number, is calculated with this law

4 First calculation : u*=0 and Nu=20

4.1 Comparisons of mean and wall temperature value at the outlet

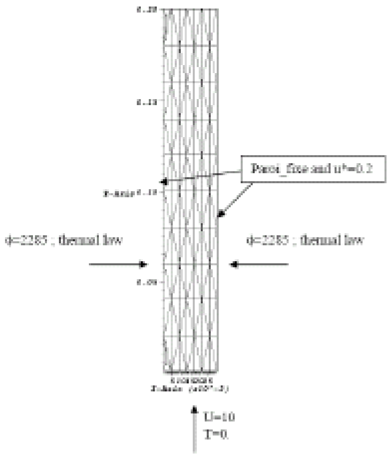
As defined before, the theorical ΔTw can be known. We show in the following table, the different values calculated by Trio_U and compare with the analytical solution.
The first ΔTw value corresponds to the difference between the first fluid temperature value and the face temperature at the wall. Thus it is disturbed by the convection term discretization.
The second ΔTw equiv value corresponds to the difference between the first fluid temperature value and the equivalent wall temperature calculated with the local Nusselt number in Trio_U.

The same calculation has be done with two different boundary conditions at the wall : 'symetrie' or 'paroi_fixe' to see the effect of non-tangential velocities at the edge

	Tmean Outlet	Tfluid	Twall (face)	ΔTw	Twall equiv	ΔTw equiv
Theorical value	0.1333	-	-	0.7201	-	0.7201
Trio_U paroi_fixe	0.1333	0.4629	0.5377	0.0747	1.182	0.7191
Trio_U symet	0.1333	0.4624	0.5375	0.0751	1.1815	0.7191

5 Second calculation : u*=0.2 and thermal law function

5.1 New Boundary conditions



5.2 Second calculation results

Here, the wall friction velocity is imposed. Thus, the wall temperature gradient is calculated with this u* value. The theorical value of ΔTw is given thanks to the Kader law. We compare it with the Trio_U results, resumed in the table below.

	Tmean Outlet	Tfluid	Twall (face)	ΔTw	Twall equiv	ΔTw equiv
Theorical value	0.1333	-	-	5.6791	-	5.6791
Trio_U paroi_fixe	0.1333	0.4629	0.5377	0.0747	6.1421	5.6791

5.3 Conclusions

We can note than the temperature gradient calculated with the Trio_U face temperature(*) is very far from the analytical one. This is due to the convection terms in the VEF discretization method.
Because of this convection effect, the modification of the local Nusselt number as no visible effect on the wall temperature.
When we take the wall equivalent temperature calculated with the thermal wall law, or the local Nusselt number (*), the ΔTw_eq is perfectly correct.
We can assure that the total wall heat is brougth to the fluid since the mean temperature at the channel outlet is correct.

(*) : this temperature values can be found in the T_paroi_pbf_Nusselt.face output file.

5.4 Computing performance

	host	system	Total CPU Time	CPU time/step	number of cell
Nu_impose/T_paroi	timber	Linux	15.9019	0.00167987	144
Symet/T_paroi	timber	Linux	15.2956	0.00166642	144
U_impose/T_paroi	timber	Linux	15.5642	0.00166959	144
Total			46.7617		

END OF DOCUMENT
FIN DU DOCUMENT