Automating CFD for non-experts

Hervé P. Morvan

ABSTRACT

The focus of the paper is on demonstrating how it is possible to automate complex CFD simulations using scripting language around and within the structure of the CFD command files. To illustrate this, the concept of an atmospheric pollution case is used and, more specifically, that of a water treatment plant. The code that is used is CFX-5 with PERL as a scripting 'language'.

The simulation of the factory atmospheric environment and its fluctuating conditions are fully automated. The simulation is based on a pre-defined generic CFD model, for which initial conditions, boundary conditions and source terms of atmospheric pollutant release are written automatically by the scripts using data recorded by measuring devices and stored on computers every half an hour as the simulation runs. When the correct amount of time has elapsed, the simulation pauses and the script updates the set-up using the newly recorded data. It then proceeds further, restarting from the appropriate result files. At each pause, a HTML report is also produced, which contains pictures of the area and summary tables. If a suitable criterion is defined in the post-treatment algorithm, such as a critical concentration for example, an alarm bell can be started, so that the technician knows the simulation has found a potential problem within the large domain that is thus monitored.

The implications of this work are numerous. Firstly, non-CFD experts can run and use results from a CFD simulation without having to implement the models, run the simulation or fully understand the intricacy of the physics and mathematics that it contains. Going further, it is even possible to parametrize the generic model set-up, e.g. the domain dimensions or the location of emission sources, to make the case more flexible. Running the application remotely is also possible, using a web browser to submit the necessary input to the CFD code. Secondly, a very wide area can be monitored numerically, which would not be commercially viable with physical devices and field monitoring campaigns. Thirdly, such a simulation can be used to learn the general behaviour of, and the potential problems associated with, the region of interest and eventually set up a response plan to any given situation known to cause discomfort or form a health hazard to the neighbourhood. This feedback can be used to improve the operation of the plant and its safety, but also to enhance the model set-up for future simulations.

Key words | atmospheric pollution, automation, CFD, real-time, simulation

Hervé P. Morvan
School of Civil Engineering,
The University of Nottingham,
Nottingham
NG7 2RD, UK
Tel: +44 115 846 6374
Fax: +44 115 951 3898
E-mail: *herve.morvan@nottingham.ac.uk*

INTRODUCTION

The on-going developments of computing facilities and CFD mean that simulations are now possible on a variety of platforms that are very efficient and affordable. Part of the progress of CFD in recent years has been its ability to become more accessible to non-specialists via the creation of graphical user interfaces (GUIs) which make the problem description faster and easier in the sense that guidelines are provided and that mutually exclusive options are automatically presented to guide the user. Although improved, this new generation of tools still

Downloaded from https://iwaponline.com/jh/article-pdf/7/1/17/392687/17.pdf
by guest

requires some CFD know-how to be used and a good knowledge of the interface to be properly operated. To an occasional end user interested in the interpretation of the results, this might constitute a barrier if he/she does not have this base knowledge or if it is not economically effective for him/her to take the time to learn and maintain this knowledge. The idea of this paper is therefore to show how the use of scripting languages could allow the modification of a pre-defined model by a non-expert user from a HTML window, for example, or even better, as will be detailed hereafter, in fully automated fashion from the use of recorded data files that feed the simulation automatically.

The CFD code that has been used is CFX-5 and it is shown how it is possible to automate the simulation of a series of atmospheric events working with a generic CFD model whose boundaries and source terms change with time as a function of live records. The scripting language used is PERL (Practical Extraction and Report Language) that most Unix users will know as a close relative of Sed or Awk, and others will recognize as a tool close to the C language. PERL's main advantage is its flexibility and ease of use, as well as the fact it is freeware (see www.PERL.com or http://www.911media.org/workshops/PERLclass/). The data from the measuring devices are recorded in plain text and can take any format since it is fairly easy to sample and reformat.

A fictitious water plant has been implemented in CFX-5. This model will serve principally to describe the geometry of the area and that of the plant and name different regions in space, such as boundary conditions or zones of emission, as well as carry the associated mesh and the general description of the gas mixture. However no prior definition of the flow dynamics and mass input will be created. This method has already been implemented by the author for a real water plant over an area of over 100 km^2. Although the example used for the figures is illustrative here, the concept has been applied and works well in practice. It should be noted that there exist other applications of interest to this forum that could be modelled in the same fashion, computer technology permitting: flood and water drainage system management could be amenable to such technology indeed, but also more general classes of chemical pollution, atmospheric or contained, or in water.

CFD – A DEMANDING TOOL WITH PROMISE

Computational Fluid Dynamics (CFD) is a demanding field for several reasons.

First of all, the Navier–Stokes equations still form one of the most difficult numerical problems to solve. This implies that a good understanding of the equations' constitution, and subsequent behaviour, may be required in order to interpret the outcome of the simulations. It also implies that a good understanding of numerical analysis is needed. In recent years various groups have tried to raise the attention of new users to these issues and have attempted to promote the concept of verification – 'solving the equations right' (Roache 1998).

From a physical standpoint, the above set of equations is not much easier to solve properly. In fact, in order to be able to solve most practical problems, models are used to approximate the behaviour of the fluid, e.g. turbulence effects. These require further physics knowledge which, combined (multiphase models, reactive flow models or radiation models), can make such problems inaccessible to all but expert users. Validation – 'solving the right equations' – then forms a second necessity.

The fact that numerical issues are often interrelated with physical ones, e.g. the boundary condition in the neighbourhood of a wall, does not make things any easier.

The progress of commercial CFD has made the above issues more accessible to new users from the standpoint of the interface, but not necessarily any easier from a fundamental perspective. Research has supplied the background technology to commercial vendors, with the sponsorships of powerful sponsors in the energy, aeronautical, chemical and car industries. It is also supplying guidelines to the implementation of CFD in industry. But CFD still requires a background knowledge whose cost is still high to industrialists for whom CFD only provides either an occasional tool or a tool that is too sophisticated to be used in its most general form in their daily business.

The author believes the role of CFD in the water industry is set to expand, which is also true in several other civil and environmental engineering sectors. CFD is a powerful tool to model geometries at the 1:1 scale and for designing new solutions interactively and rapidly. It is also becoming an economical tool in terms of hardware, with

Downloaded from https://iwaponline.com/jh/article-pdf/7/1/17/392687/17.pdf
by guest

the advent of the PC and Linux, and software. What is expensive to buy and maintain is the human CFD knowledge. But is that necessary to all businesses? If CAE (Computer Aided Engineering) and CFD do not constitute an essential part of the business and the design of new solutions is only occasional, the answer is no. Should it prevent these industries from using CFD, e.g. for operational purposes? The author believes the answer is no once again and the water industry should, in fact, become one of these CFD clients for whom the developments shown in this paper could be directly applicable.

The concept of *hydroinformatics* is a combination of CAE and modern communication technology aimed at making engineering models and information accessible, and assisting in decision-making. Scripting, and embedded scripting in particular, implies that it is now possible for an expert user to prepare generic model set-ups that are subsequently usable by a wider range of users who are experts in the application field, but not in CFD for example. Going further, it is even possible to generate 'smart' applications that can run automatically using information directly obtained from recorded devices, such as wind or water meters. These smart applications can run rapidly and generate easy-to-distribute web-based reports. This is what is presented below: CFD made accessible via the implementation of PERL scripts that manage the simulations and produce legible results automatically.

Expert users create the model, write the scripts and supply a simple front-end to end users. The latter can then fill in the data and submit jobs that can run remotely, for example on sites dedicated to providing computer power (CPU time) and access to the necessary software licenses. All of this can form part of a service that does not necessarily need to be part of the end user organization. It can therefore be very cost-effective, especially for small firms.

CFX-5

The code CFX-5 has been tested for various environmental applications and has been found to be suited for this application. The concepts that are presented hereafter should, however, be applicable to other CFD codes.

CFX-5 is a fully unstructured code able to tackle complex geometries. Its solver is particularly efficient and relies on a fully coupled approach and an algebraic multigrid acceleration technique. Another attraction of this code is that it encompasses a breadth of physical models that are particularly useful to water and environmental applications such as free surface and general Eulerian–Eulerian multiphase models for example. These models can be further developed using FORTRAN routines available to expert users.

Within the context of the present paper, however, the most interesting asset of the code is the fact that it is able to interpret PERL directly. This implies that it is possible to write conditional CFX commands using embedded PERL. As a PERL interpreter[1] is provided it is then possible to use PERL to manage the files and the simulation 'externally' using batches and to drive the CFX-5 simulation 'internally' from inside the definition file.[2]

APPLICATION

The main objective is to monitor the dispersion of heavy gas, such as H_2S for example, away from emission zones: open water and mud treatment tanks of the water treatment plant. The problem with such gas is that it is extremely nauseous, even in relatively small concentrations, and that it is the duty of the operator to ensure that such emission is properly timed so as to cause minimum discomfort to the neighbourhood; something that is crucial in an urban or semi-urban environment.

The interests of the operator are:

1. To manage its operation.
2. To improve its understanding of the region (which is impossible using a monitoring campaign carried out from a mobile laboratory on such a scale).
3. To create a response plan, ahead of time, to various meteorological and venting situations with time.

[1]PERL is not compiled; it is therefore not a language *per se* but rather forms a series of script commands. Such an interpreter can be downloaded free of charge on the web.
[2]The definition file is the CFX-5 file containing the mesh and all the commands necessary to run the simulation.

Downloaded from https://iwaponline.com/jh/article-pdf/7/1/17/392687/17.pdf
by guest

4. To possess a supporting tool used to answer queries from the public and the local authorities.
5. To show its willingness to control and limit the possible discomfort caused by the treatment process by improving the management and understanding of the operation.

The main constraints are that:

1. The application end users have little to no modelling knowledge.
2. The tool therefore needs to be automated and open:
   (a) It needs to read and sample recorded data for the proper flow profile values at the inlets at every time interval.
   (b) It needs to detect the flow direction to create boundary types (inlets, outlets, openings) as it changes with time.
   (c) It needs to read and sample measured emission data to add pollutant sources where necessary at every time interval.
   (d) It needs to stop and be able to restart with updated conditions from the correct time period, at every time interval, in a reliable fashion.
   (e) It needs to be evolutive so that it can quickly reflect changes in the operation (e.g. new sources).
3. The whole simulation has to happen in 'real time' (the simulation actually runs with a slight delay due to the fact that data need to be recorded before they can be used).

On the other hand the simulation domain is fixed and well defined. Therefore the geometry and the mesh can be generated once by a CFD specialist at the start of the project and eventually enhanced, if required, when feedback is available from end users. It may be advisable to ask the CFD expert for a periodic review of the model to ensure that all is well with the set-up, e.g. on a yearly basis.

The case presented here is just an example, of course: such a concept could be applied to various other applications with CFD codes or others. The PERL scripts developed by the author could be easily adapted to other cases. However, the constraints listed above are believed to be representative of most applications that could be attracted to such technology.

CREATION OF THE CFX-5 GENERIC MODEL

Topographical data for the area are provided in text format in X, Y, Z columns. A simple script is written to convert these data to Patran format, the underlying language in the CFX current pre-processor, so that the necessary points and lines constituting the bottom topographical surface can be turned automatically into a CFX-5 geometry in the CFX-Build, CFX CAD and meshing facility. An example of a script written for this purpose is given in Appendix 1.

Once the surface is created and smoothed in CFX-Build, it is fairly straightforward to construct the sides and top of the box forming the simulation domain. Here the top surface is called the 'Sky', and the box's sides named 'East', 'North', 'West' and 'South'. Various patches[3] based on geographical and physical properties are also created at this stage on the bottom surface so as to be able to implement a suitable roughness representation for water, open fields, woods and forest, housing estates, town and tall building areas or estates. Roughness values can be found in the literature (Aynsley 1977) for each of these.

The above computational box is then meshed and a generic physical set-up is implemented in CFX-Pre, the CFX-5 physics pre-processor. The idea behind the concept that is presented here is that additional files called `BCs`[4]`.ccl`, `InputBCs.ccl` and `InputPollutant.ccl`, standing for boundary condition type, boundary condition values and source term values respectively, will be created by the automated script from the recorded data and added to the CFX definition file.[5] These new pieces of information will overwrite pre-defined sections, where appropriate, in the generic definition file. This is done automatically whenever necessary. The file extension `ccl`

[3]Patches is the name used in CFD to refer to groups of computational nodes contained on a surface forming a boundary of the simulation domain.
[4]BC stands for Boundary Condition; BCs is the plural form.
[5]A definition file is the file that contains all the necessary information (domain, mesh, physics and numerical options) required to define the problem and run a CFD solver.

Downloaded from https://iwaponline.com/jh/article-pdf/7/1/17/392687/17.pdf
by guest

stands for CFX Command Language, but in reality these files are simply additional text files of keywords that are concatenated to the generic definition file as appropriate.

The reader should note that the end user does not need to understand the detail of the CFX Command Language as the appropriate definition file is constructed by the algorithm. This is just presented for clarity and for the benefit of those who would like to create their models using a similar approach.

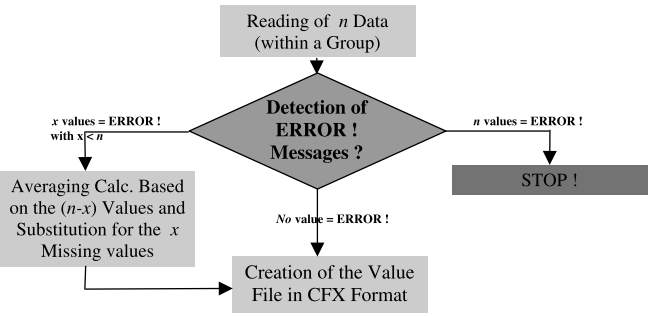

Figure 1 | Data sampling for `InputBCs.ccl` and `InputPollutant.ccl`.

ALGORITHM

Reading and sampling the measured data

The live data must be read and formatted so that they can be used by CFX. These data are the wind direction and intensity with altitude as well as the emission of products from the plant (sources). In an ideal situation, the data files are perfect and contain no errors, and measuring devices are never out of order or being serviced. Of course, this is not the case; therefore one must be able to detect missing data and errors and have a protocol to report and/or substitute them. It is decided that by default the error message that is reported for a missing or erroneous data is `ERROR!` It is followed by information regarding the location of the data in the file and suggestions to alleviate the problem when no automated solution is applicable. A typical way to substitute a missing wind measurement automatically is to look at the neighbouring measurement values and the previous time series and use these to extrapolate a new value; for a source, a mean value can be computed using other source measurements.

To complicate data management for the sources further, the grouping of some data is necessary to reflect different zones of pollutant emission in the plant. In the event where a source data point is missing, e.g. due to a failing captor, it can then be substituted by a representative average of its neighbours within the zone. The other source points located outside the zone are not used in the computation of the substitute value. Groups of data therefore need to be sampled separately. Which data belongs to which group is known from the start (and defined as such in the algorithm) and the chart below aims to illustrate the sampling of n values belonging to one group of data.

Creating the boundary types

Once the data file has been created it is also necessary to detect the flow direction so as to be able to determine which of the boundaries, West, North, East or South, will be defined as an inlet and which will become a static pressure or outlet boundary condition to reflect an outflow condition. This is easily done by looking at the signs of the numerical values within the frame of reference chosen for the simulation, once the measured values have been formatted in this frame of reference.

Since several wind values can be recorded at various measuring poles it is possible to represent a piecewise linear wind profile with altitude. This information is therefore used to create a 2D profile for the wind, based on a pre-defined parametric profile that is implemented using the measured values sampled as shown in Figure 1 for example. Values from the various masts can then be smoothed and implemented in the correct locations, West, North, East or South, depending on the wind direction, see Figure 2.

At this stage of the simulation, all the fluctuating boundaries and source data have been accounted for and formatted as CFX-5 data files. What is required is to run the simulation in automated batch mode.

Management of the simulation and run in batch mode

The algorithm presented above is run at every time interval, i.e. the time period corresponding to the elapsed time

Downloaded from https://iwaponline.com/jh/article-pdf/7/1/17/392687/17.pdf
by guest

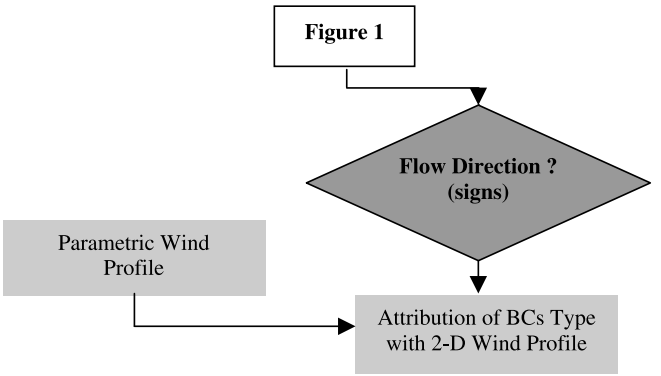

Figure 2 | Boundary condition type and 2D wind profile.

between two consecutive recordings of wind flow and pollutant emission. For any given day j and time interval $h+1$, a directory is created as follows: `Dayj/Timeh+1`, with $1 \leq j \leq 31$ and $0 \leq h \leq 47$ (assuming the time interval is equal to 30 min, therefore requiring 48 time intervals).

During the run, when a simulation ends for period `Dayj/Timeh+1` it has to restart from `Dayj/Timeh+2`, using the correct `BCs.ccl, InputBCs.ccl` and `Input-Pollutant.ccl` files produced for time interval $h+2$ and the result file from the ending simulation that is located in `Dayj/Timeh+1`, which turns into the initial condition file at `Dayj/Timeh+2`. Similarly the scripts need to manage the change between day j and day $j+1$.

In any given period, the script writes and executes a CFX-5 solver command from the right location that reads:

```
cfx5solve -def $dir/GenericDefinitionFile.def
-ccl
$dir/Day$mm/Time$m/inputPoluttant.ccl -ccl
$dir/Day$mm/Time$m/inputBCs.ccl -ccl
$dir/Day$mm/Time$m/BCs.ccl -ccl
$dir/Day$mm/Time$m/time.ccl -ccl
$dir/sourcepoint.ccl
```

where `$dir` indicates the working directory, `$mm` is the day and `$m` the time interval, all of which are replaced by the real path and values, as indicated above, as the runs iterate. The files behind the `-ccl` commands overwrite the information contained in the CFX-5 generic executable definition file with the up-to-date information.

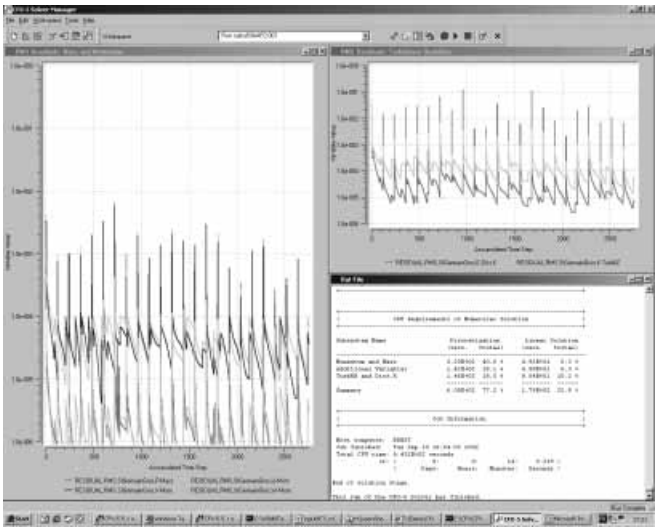

Figure 3 | Transient simulation showing convergence to 10^{-4} at each time iteration (reduction of four orders of magnitude for each time interval) for all components.

The attentive reader will notice two extra files called `time.ccl` and `sourcepoint.ccl` at the end of the command line. They have been included for flexibility to set the time interval as required, and to define the source point types and locations respectively.

Again all the items from the second to the last line of commands are generated automatically and the command is subsequently executed. The simulation will only stop when one presses the 'Stop' button in the CFX-5 GUI, Figure 3, when one reaches the end of the pre-defined total simulation time or when the data runs out!

This part of the script is shown in greater detail in Appendix 2. This type of script can be operated from a HTML window used for the definition of the parameter values.

The application described so far is working well in terms of the quality of the CFD, as the following convergence windows shows, Figure 4.

Post-treatment

To complete the work presented so far, an automated post-treatment is required, in a format that is easily usable and portable.

Downloaded from https://iwaponline.com/jh/article-pdf/7/1/17/392687/17.pdf
by guest

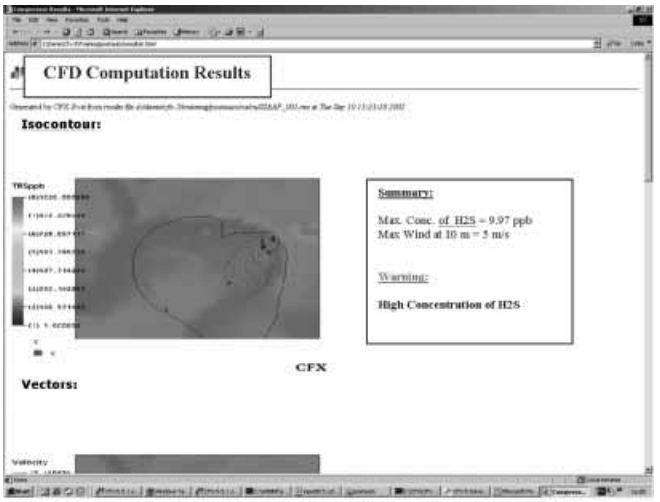

Figure 4 | Example of a simple HTML report.

Once a view plane and its associated picture type (vectors, iso-contours) are retained by the end user, they are created generically in the post-processor, here CFX-Post, and assuming that the chosen post-treatment tool can function in batch mode, it is then very simple to offer the same automation to the user as that described until now for the pre-processor and the solver. CFX-Post has such a capability, and after review with the plant operator it is then possible to prepare generic plan views at given heights, with plots of vectors and iso-contours that are systematically created at the end of each run.

Using PERL one can create JPG or PS pictures, for example, which are placed in a HTML file with dates and a summary of weather and emission conditions. Important data can be formatted in tables and an alarm bell system can be set up to warn the end user of a potential problem. This way one does not need to use the post-processor to obtain a concise summary of the latest situation.

The advantage of using a HTML automated report is that the latter can then be distributed across all platforms equipped with an internet browser such as Netscape or Internet Explorer, making the results available to anyone on the network. This has a very strong potential in terms of the distribution of the information within a company and its posting on a web site to inform the wider population of a situation.

GENERALIZATION OF THE CONCEPT

The work that has been presented here in the context of a water treatment plant could well have been applied to any other type of plant. It would even be easier from a modelling standpoint to implement the concept in an indoor situation, e.g. to monitor gas emission or (nuclear) radiation. Moving away from the application context shown above, but remaining in the water engineering industry, it would also be possible to implement generic models for a section of rivers or canals or sewage, with inlets and outlets to an installation which could be defined with parametric dimensions and physical conditions.

This paper shows that it is possible for an expert in CFD to prepare nearly any generic model that can be tailor-used by end users whose main knowledge lies in the exploitation of the fluid mechanics results and the associated mechanical or chemical processes rather than the science of CFD. PERL, as with many web and shareware tools, is also very accessible and can easily be used in conjunction with general web browsers. Since most scientific codes can be operated in batch mode, such an implementation is relatively easy and could be extended to non-CFD codes.

This work also implies that the end user could use remote resources (either hardware and/or software), e.g. in a central office or from a service provider, accessed via a web browser. In this business model we would have an expert consultant preparing the model, a service provider specialized in scientific computation and a third-party client hiring à la carte computer and applications facilities.

CONCLUSIONS

It has been shown that it is now possible to combine and customize complex CFD models with scripting languages. Tools such as PERL can serve to facilitate access to scientific codes such as CFX and also to automate simulations using remote data as direct inputs. It has also been said that other CAE tools could be amenable to such automation.

Downloaded from https://iwaponline.com/jh/article-pdf/7/1/17/392687/17.pdf
by guest

A water treatment plant has served to illustrate the concept and it has been used to demonstrate that the possibility for such combined automated simulation is extremely wide. A variety of other applications could be implemented using the same technology in the water business and beyond.

In the context of the present paper however, it implies that CFD technology could become more readily usable by a larger community within a given organization, for which an expert would create the base model and the tailored interfaces, and ensure regular maintenance. Going further this also opens the possibility for a remote use of CFD resources via the web.

ACKNOWLEDGEMENTS

The author is indebted to Mr Denis Lécuyer, formerly of AEAT France and currently working for CETIM in Nantes, France, who participated in the writing of some of the original PERL scripts that led to those shown in the appendices. I would also like to thank my colleagues at Nottingham and the reviewers for their comments on this paper.

REFERENCES

Aynsley, R. M., Melbourne, R. & Vickery, B. J. 1977 *Architectural Aerodynamics*. Applied Science, London.

Roache, P. J. 1998 *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, Albuquerque, NM.

Downloaded from https://iwaponline.com/jh/article-pdf/7/1/17/392687/17.pdf
by guest

APPENDIX 1. AUTOMATION OF TOPOGRAPHICAL SURFACE CREATION FOR CFX-BUILD

```
#################################################################################################
#                                                                                               #
#    PERL Script used to Format DTM Points in Patran Format for CFX-5 (Session files)           #
#    Author: Hervé Morvan, 2002                                                                 #
#                                                                                               #
#################################################################################################


#-----------------------------------------------------------------------------------------------#
#THIS SCRIPT WILL CREATE A PARAMETRIC SURFACE FROM ANY REGULAR MATRIX OF XYZ DATA                #
#-----------------------------------------------------------------------------------------------#
#                                        PARAMETRES                                             #
#-----------------------------------------------------------------------------------------------#
#THE ONLY FIELDS TO BE CHANGED ARE $a, $b, THE PATTERN OF THE SPLIT COMMAND IF IT IS
#NOT A WHITE SPACE SEPARATOR AND THE XYZ POINT FILE NAME AND PATH
# - HPM, 06/09/02

#PLEASE RETURN COMMENTS AND MODIFICATIONS TO THE AUTHOR AS REQUIRED


[...]


##############################################
#3.                                          #
#   - Writing the chained lines              #
##############################################

$o=1;

#Matrix of Points p*q (here 10 rows×20 lines for example)
#Set values for p and q
$p=$a;
$pc=$p-1;
$pp=$p;
$ppc=$pc;
$q=$b;
$nb=($p-1)*$q;

open (WRITING,''>>surface.ses'');
print WRITING ''INTEGER sgm_create_curve_cha_segment_id\n'';
print WRITING ''STRING sgm_create_curve_ch_created_ids[VIRTUAL]\n'';

for ($n=($nb+1); $n<($nb+1+$q); $n++) {
open (WRITING, ''>>surface.ses'');

#Patran chain curves

print WRITING ''sgm_create_curve_chain_v1(\''$n\'', \''Curve $o:$pc\'', FALSE,
sgm_create_curve_cha_segment_id, sgm_create_curve_ch_created_ids)\n'';
close (WRITING);
$o=$pc+1;
$pc=$pc+$ppc;
}
```

Downloaded from https://iwaponline.com/jh/article-pdf/7/1/17/392687/17.pdf
by guest

```
##############################################
#4.                                          #
#  - Creating the topographical surface      #
#    N.B. Patran=>Nb lines<=100!             #
##############################################

#Creation of session file ''surface.ses''
open (WRITING, ''>>surface.ses'');

print WRITING ''STRING sgm_surface_ncurve_created_ids[VIRTUAL]\n'';

$nbb=$nb+1;
$nbbq=$nbb+$q-1;

print WRITING ''sgm_const_surface_ncurve_v1(\''1\'', 2, \''Curve $nbb:$nbbq\'',
sgm_surface_ncurve_created_ids)\n'';
```

Downloaded from https://iwaponline.com/jh/article-pdf/7/1/17/392687/17.pdf
by guest

APPENDIX 2. PERL SCRIPT (PART)

```
#################################################################################################
#                                                                                               #
#    PERL Script                                                                                #
#    Authors: Hervé Morvan and Denis Lécuyer (currently at CETIM, France)                       #
#                                                                                               #
#                                                                                               #
#################################################################################################

require ''ParametresPERL.txt'';

#################################################################################################
#    MAIN PROGRAMME                                                                             #
#################################################################################################

#EXECUTION OF ROUTINES

cclconcentration ($dir1,$myfile1);
cclmeteo ($dir1,$myfile2);
ccltemps($dir1,$totaltime1);
calcbatch($dir1,$initialday1,$endday1,$testinitres1,$initres1);

#-----------------------------------Main Programme End-----------------------------------#

[...]

#################################################################################################
#################################################################################################
#    SUB-ROUTINE PERL to launch a series of runs in batch mode                                  #
#################################################################################################
#################################################################################################

sub calcbatch {
($dir,$initialday,$endday,$tEastinitres,$initres)=@_;
print ''STARTING SIMULATION\n'' ;

#############################################
#1. LOOP ON DAYS                            #
#############################################

for ($mm=$initialday; $mm<=$endday; $mm++) {
print ''CALCULATING DAY:$mm\n'' ;

#############################################
#2. LOOP ON ALL TIMES INTERVALS OF ONE DAY   #
#############################################

for ($m=0; ($m<=47); $m++) {
print ''CALCULATING DAY: J$mm TIME:$m\n'' ;
```

Downloaded from https://iwaponline.com/jh/article-pdf/7/1/17/392687/17.pdf
by guest

```
##################
#2.1 CALCULATION #
##################
$mmc=$mm;
$mc=$m-1;
if ($m==0) {

              $mc=47;
              $mmc=$mm-1;

}

#Working directory (directory from which the CFX run is started)
chdir (``$dir/day$mm/time$m'');

#Initialization :
#no initial file
if (($mm==$initialday)\($m==0)\($tEastinitres==0)) {
open(SOLVE,`` | cfx5solve -def $dir/calculSIAAP.def -ccl $dir/pointsource.ccl -ccl
$dir/day$mm/time$m/inputH2S.ccl -ccl $dir/day$mm/time$m/inputBCS.ccl -ccl
$dir/day$mm/time$m/BCS.ccl -ccl $dir/day$mm/time$m/temps.ccl'');
close(SOLVE); # start the run going
}
#if initial file available
if (($mm==$initialday)\($m==0)\($tEastinitres!=0)) {
open(SOLVE,'' | cfx5olve -def $dir/calculSIAAP.def -ini $initres -ccl
$dir/pointsource.ccl -ccl $dir/day$mm/time$m/inputH2S.ccl -ccl $dir/day$mm/time$m/inputBCS.ccl
-ccl $dir/day$mm/time$m/BCS.ccl -ccl $dir/day$mm/time$m/temps.ccl'');
close(SOLVE); # start the run going
}

#Calculation of the following time intervals
if (($mm!=$initialday) | | (($mm==$initialday)\($m!=0))) {
open(SOLVE,'' | cfx5solve -def $dir/calculSIAAP.def -ini $dir/day$mmc/time$mc/calculSIAAP_001.
res -ccl $dir/pointsource.ccl -ccl
$dir/day$mm/time$m/inputH2S.ccl -ccl $dir/day$mm/time$m/inputBCS.ccl -ccl
$dir/day$mm/time$m/BCS.ccl -ccl $dir/day$mm/time$m/temps.ccl'');
close(SOLVE); # start the run going
}

}#end of loop on time intervals

}#end of loop on days

#end of routine routine calcbatch
```

Downloaded from https://iwaponline.com/jh/article-pdf/7/1/17/392687/17.pdf
by guest

Key:

```
CFX commands.
PERL commands.
Calcbatch subroutine, called in the body of the main programme.
```

Downloaded from https://iwaponline.com/jh/article-pdf/7/1/17/392687/17.pdf
by guest