TelApy

User Manual

Version v8p5
December 1, 2023

Contents

1 Introduction . . . 3

2 TelApy description . . . 5

2.1 Telemac system Fortran API description 5

2.1.1 Structure . . . 6

2.1.2 Instantiation . . . 6

2.1.3 Variable control . . . 15

2.1.4 Computation control . . . 15

2.1.5 Parallelisation . . . 16

2.2 TelApy Python module 16

3 Getting Started with TelApy module . . . 17

3.1 TelApy module installation 17

3.2 How to run notebook documentation 17

3.3 Notebook examples in TelApy 17

4 Developer manual . . . 19

4.1 Instance 19

4.1.1 Instance functions . . . 20

4.1.2 How to add access to a new variable . . . 20

4.2 Coding conventions 21

4.2.1 Fortran part of API . . . 21

4.2.2 TelApy module . . . 21

4.3 Validation 22

5 Outlooks . . . 23

Bibliography . . . 24

1. Introduction

This guide aims to explain the use of the TELAPY module of the TELEMAC SYSTEM. This module aims to provide python control of TELEMAC API (Application Program Interface).
The API's main goal is to have control on a simulation while running a case. For example, it must allow the user to stop the simulation at any time step, retrieve some variable values and change them. In order to make this possible, a Fortran structure called instance is used in the API. It contains a list of variables declared as pointers that are pointing to variables. This gives direct access to the physical memory of variables, and allows therefore to retrieve their values, and modify them. Furthermore, based on modifications in TELEMAC main subroutines, it is possible to run hydraulic case time step by time step.
Thus, the APIs development allows the interoperability of the TELEMAC SYSTEM modules. Interoperability is the ability of a computer system to operate with other existing or future informatic products without restricting access or implementation. It, then, becomes natural to drive these APIs using Python programming language. In fact, Python is a portable, dynamic, extensible, free language, which allows (without imposing) a modular approach and object oriented programming. Python has been developed since 1989 by Guido van Rossum and many

volunteer contributors. In addition of the benefits of this programming language, Python offers a large amounts of interoperable libraries. The link between various interoperable libraries with TELEMAC SYSTEM APIs allows the creation of an ever more efficient computing chain able to more finely respond to various complex problems.
Consequently, the TELAPY module has the ambition to facilitate the usage of TELEMAC SYSTEM for optimization, coupling, uncertainty quantification… applications.

2. TELAPY description

As mentioned in the introduction part (Section 1), the TELAPY module is used to control the APIs of TELEMAC SYSTEM in the Python programming language. The TELEMAC SYSTEM APIs are developed in Fortran. However, it is relatively easy to use these Fortran routines directly in Python using the “f2py” tool from the Python NumPy library [1]. This tool will make it possible to compile Fortran code to make it accessible and usable in Python. This compilation step is directly integrated into the compilation of the TELEMAC SYSTEM and is thus transparent to the user. Moreover, in order to make the TELAPY module more user friendly, a Python wrapper has been developed in order to encapsulate and simplify the different API Python calls. This set of transformation constitutes the TELAPY module.
The first section of this chapter is dedicated to the Fortran API of TELEMAC SYSTEM. Then, the Python TELAPY module is presented.

2.1 TELEMAC SYSTEM Fortran API description

An API (Application Programming Interface) is a library allowing to control the execution of a program. Here is part of the definition from Wikipedia:

"In computer programming, an application programming interface (API) specifies a software component in terms of its operations, their inputs and outputs and underlying types. Its main purpose is to define a set of functionalities that are independent of their respective implementation, allowing both definition and implementation to vary without compromising each other.
In addition to accessing databases or computer hardware, such as hard disk drives or video cards, an API can be used to ease the work of programming graphical user interface components, to allow integration of new features into existing applications (a so-called "plug-in API"), or to share data between otherwise distinct applications. In practice, many times an API comes in the form of a library that includes specifications for routines, data structures, object classes, and variables."

The API’s main goal is to have control on a simulation while running a case. For example, it must allow the user to stop the simulation at any time step, retrieve some variables values and change them. In order to make this possible, a Fortran structure called instance is used in the API. This informatic structure is described in the paragraph 2.1.2. The instance structure gives direct access to the physical memory of variables, and allows therefore the variable control (see paragraph 2.1.3). Furthermore, based on modifications in TELEMAC SYSTEM main subrou-

tines, it is possible to run hydraulic case time step by time step. This will be presented in the paragraph 2.1.4. And finaly, the parallelism is evoked (paragraph 2.1.5). All Fortran routines are available in the directory “api” of the TELEMAC SYSTEM repository. sources.
Detailed information on each function can be find in the Doxygen documentation to open it run the following command:

```
firefox <root>/documentation/doxydocs/html/index.html
```

Replacing <root> by the path to your installation and firefox by your internet browser.
If you are on the main branch, run the command below to generate it.

```
doc_telemac.py -M doxydocs
```

Warning:
In the following sections, all presented API routines are related to TELEMAC-2D. However, the API implementation of TELEMAC SYSTEM modules is generic that is to say based on the same structure (in the following routines, the sequence "t2d" related to TELEMAC-2D module can be replaced e.g. with "t3d" to use the TELEMAC-3D module).

2.1.1 Structure

The functions are dispatched in Fortran files as follow:

- `api_interface.f`: Contains the main function for all the api.
- `api_handle_error.f`: Contains the handling of error messages.
- `api_handle_var_t2d.f`: Contains all the get/set.
- `api_instance_t2d.f`: Contains the function for the instance described in Section 2.1.2.
- `api_run_t2d.f`: Contains the function for the execution of the computation described in Section 2.1.4.

2.1.2 Instantiation

An instance is a memory structure that gathers all the variables alterable by the API. The definition of the “instance” structure is made in a Fortran type dedicated to this purpose and is composed of:

- An index defining the instance ID.
- A string which can contain error messages.
- Some pointers to the concerned module variables. This is what makes it possible to manipulate the variables of the module by having a direct access to their memory location. The list of variables that can be accessed is given in the Tables below. Not all variables within the module are there. However adding a new variable is pretty straight forward, the procedure is described in Section 4.1.2.

Variable name	Definition

MODEL.AT	CURRENT TIME
MODEL.DT	TIME STEP
MODEL.TMAX	END TIME
MODEL.BCFILE	BOUNDARY CONDITION FILE NAME
MODEL.BND_TIDE	OPTION FOR TIDAL BOUNDARY CONDITIONS
MODEL.BOTTOMELEVATION	LEVEL OF THE BOTTOM
MODEL.CHESTR	ROUGHNESS COEFFICIENT ON POINT
MODEL.FAIR	FAIR ON POINT
MODEL.COTE	PRESCRIBED ELEVATION ON BOUNDARY
MODEL.DEBIT	PRESCRIBED DISCHARGE ON BOUNDARY
MODEL.DEBUG	ACTIVATING DEBUG MODE
MODEL.FLUX_BOUNDARIES	FLUX AT BOUNDARIES
MODEL.GEOMETRYFILE	NAME OF THE GEOMERY FILE
MODEL.METEOFILE	NAME OF THE BINARY ATMOSPHERIC FILE
MODEL.FO2FILE	NAME OF THE FORMATTED DATA FILE 2
MODEL.LIQBCFILE	NAME OF THE LIQUID BOUNDARIES FILE
MODEL.PREFILE	NAME OF THE PREVIOUS COMPUTATION FILE
MODEL.GRAPH_PERIOD	GRAPHICAL OUTPUT PERIOD
MODEL.HBOR	BOUNDARY VALUE ON H FOR EACH BOUNDARY POINT
MODEL.IKLE	CONNECTIVITY TABLE BETWEEN ELEMENT AND NODES
MODEL.ELTSEG	SEGMENTS FORMING AN ELEMENT
MODEL.ORISEG	ORIENTATION OF SEGMENTS FORMING AN ELEMENT
MODEL.GLOSEG	GLOBAL NUMBERS OF VERTICES OF SEGMENTS
MODEL.NACHB	NUMBERS OF PROC CONTAINING A GIVEN POINT
MODEL.KNOLG	GIVES THE INITIAL GLOBAL NUMBER OF A LOCAL POINT
MODEL.INCWATERDEPTH	INCREASE IN THE DEPTH OF THE WATER
MODEL.KP1BOR	POINTS FOLLOWING AND PRECEDING A BOUNDARY POINT
MODEL.LIHBOR	BOUNDARY TYPE ON H FOR EACH BOUNDARY POINT
MODEL.LISTIN_PERIOD	LISTING OUTPUT PERIOD
MODEL.LIUBOR	BOUNDARY TYPE ON U FOR EACH BOUNDARY POINT
MODEL.LIVBOR	BOUNDARY TYPE ON V FOR EACH BOUNDARY POINT
MODEL.LT	CURRENT TIME STEP
MODEL.NBMAXNSHARE	MAXIMUM GEOMETRICAL MULTIPLICITY OF A NODE
MODEL.COMPLEO	GRAPHIC OUTPUT COUNTER
MODEL.PTINIG	NUMBER OF FIRST TIME STEP FOR GRAPHIC PRINTOUTS
MODEL.NPTIR	NUMBER OF INTERFACE POINTS OF THE SUBDOMAIN
MODEL.NBOR	GLOBAL NUMBER OF BOUNDARY POINTS

MODEL.NELEM	NUMBER OF ELEMENT IN THE MESH
MODEL.NELMAX	MAXIMUM NUMBER OF ELEMENTS ENVISAGED
MODEL.NPOIN	NUMBER OF POINT IN THE MESH
MODEL.NSEG	NUMBER OF SEGMENT IN THE MESH
MODEL.NPTFR	NUMBER OF BOUNDARY POINTS
MODEL.NTIMESTEPS	NUMBER OF TIME STEPS
MODEL.NUMLIQ	LIQUID BOUNDARY NUMBERS
MODEL.POROSITY	POROSITY
MODEL.RESULTFILE	NAME OF THE RESULT FILE
MODEL.SEALEVEL	COEFFICIENT TO CALIBRATE SEA LEVEL
MODEL.TIDALRANGE	COEFFICIENT TO CALIBRATE TIDAL RANGE
MODEL.VELOCITYRANGE	COEFFICIENT TO CALIBRATE TIDAL VELOCITY RANGE
MODEL.UBOR	BOUNDARY VALUE ON U FOR EACH BOUNDARY POINT
MODEL.VBOR	BOUNDARY VALUE ON V FOR EACH BOUNDARY POINT
MODEL.VELOCITYU	VELOCITY ON U
MODEL.VELOCITYV	VELOCITY ON V
MODEL.WATERDEPTH	WATER DEPTH
MODEL.WATERDEPTHN	WATER DEPTH AT PREVIOUS TIME STEP
MODEL.X	X COORDINATES FOR EACH POINT OF THE MESH
MODEL.XNEBOR	NORMAL X TO 1D BOUNDARY POINTS
MODEL.Y	Y COORDINATES FOR EACH POINT OF THE MESH
MODEL.YNEBOR	NORMAL Y TO 1D BOUNDARY POINTS
MODEL.EQUATION	NAME OF THE EQUATION USED
MODEL.AK	TURBULENT KINETIC ENERGY K
MODEL.EP	TURBULENT DISSIPATION EPSILON
MODEL.ITURB	TURBULENCE MODEL
MODEL.INIT_DEPTH	INITIAL DEPTH
MODEL.TRACER	TRACERS VALUE
MODEL.NTRAC	NUMBER OF TRACERS
MODEL.FLOWRATEQ	SOLID TRANSPORT FLOWRATE
MODEL.DCLA	MEDIAN GRAIN SIZE
MODEL.SHIELDS	CRITICAL SHIELDS PARAMETER
MODEL.XWC	SETTLING VELOCITY
MODEL.Z	FREE SURFACE ELEVATION
MODEL.QBOR	BOUNDARY VALUE ON Q FOR EACH BOUNDARY POINT
MODEL.EBOR	BOUNDARY VALUE ON E FOR EACH BOUNDARY POINT
MODEL.FLBOR	BOUNDARY VALUE ON ZF FOR EACH BOUNDARY POINT
MODEL.TOB	SHEAR STRESS
MODEL.CLU	BOUNDARY TYPE ON U FOR EACH BOUNDARY POINT
MODEL.CLV	BOUNDARY TYPE ON V FOR EACH BOUNDARY POINT

MODEL.LIQBOR	BOUNDARY TYPE ON Q FOR EACH BOUNDARY POINT
MODEL.LIEBOR	BOUNDARY TYPE ON E FOR EACH BOUNDARY POINT
MODEL.NSICLA	NUMBER OF SIZE-CLASSES OF BED MATERIAL
MODEL.NOMBLAY	NUMBER OF LAYER IN THE BED
MODEL.CONCENTRATION	CONCENTRATION AT TIME N
MODEL.EVOLUTION	EVOLUTION OF BED
MODEL.PARTHENIADES	PARTHENIADES EROSION COEFFICIENT FOR EACH BED LAYER FOR EACH CLASS
MODEL.MARDAT	A 1 DIMENSIONAL INTEGER ARRAY
MODEL.MARTIM	A 1 DIMENSIONAL INTEGER ARRAY
MODEL.RAZTIM	A SIMPLE BOOLEAN
MODEL.START_RECORD	A SIMPLE INTEGER
MODEL.VOLU2D	INTEGRAL OF BASES DOUBLE BIEF_OBJ
MODEL.AM1D	DIAGONAL OF AM1 MATRIX
MODEL.AM1X	EXTRA-DIAGONAL TERMS OF AM1 MATRIX
MODEL.AM2D	DIAGONAL OF AM2 MATRIX
MODEL.AM2X	EXTRA-DIAGONAL TERMS OF AM2 MATRIX
MODEL.AM3D	DIAGONAL OF AM3 MATRIX
MODEL.AM3X	EXTRA-DIAGONAL TERMS OF AM3 MATRIX
MODEL.BM1D	DIAGONAL OF BM1 MATRIX
MODEL.BM1X	EXTRA-DIAGONAL TERMS OF BM1 MATRIX
MODEL.BM2D	DIAGONAL OF BM2 MATRIX
MODEL.BM2X	EXTRA-DIAGONAL TERMS OF BM2 MATRIX
MODEL.CM1D	DIAGONAL OF CM1 MATRIX
MODEL.CM1X	EXTRA-DIAGONAL TERMS OF CM1 MATRIX
MODEL.CM2D	DIAGONAL OF CM2 MATRIX
MODEL.CM2X	EXTRA-DIAGONAL TERMS OF CM2 MATRIX
MODEL.A32D	DIAGONAL OF A32 MATRIX
MODEL.A32X	EXTRA-DIAGONAL TERMS OF A32 MATRIX
MODEL.A23D	DIAGONAL OF A23 MATRIX
MODEL.A23X	EXTRA-DIAGONAL TERMS OF A23 MATRIX
MODEL.CV1	TERMS OF CV1 VECTOR
MODEL.CV2	TERMS OF CV2 VECTOR
MODEL.CV3	TERMS OF CV3 VECTOR
MODEL.PROPNU	DIFFUSION COEFFICIENT OF VELOCITY
MODEL.MPM	MEYER PETER MUELLER-COEFFICIENT
MODEL.BETA2	PARAMETER FOR DEVIATION
MODEL.PHISED	FRICTION ANGLE OF THE SEDIMENT
MODEL.ALPHA	SECONDARY CURRENT ALPHA COEFFICIENT
MODEL.XKV0	INITIAL POROSITY BY LAYERS
MODEL.KSPRATIO	RATIO BETWEEN SKIN FRICTION AND MEAN DIAMETER
MODEL.IORDRH	INITIAL GUESS FOR DEPTH
MODEL.IORDRU	INITIAL GUESS FOR VELOCITY
MODEL.INCVELOCITYU	INCREASE OF VELOCITY U
MODEL.INCVELOCITYV	INCREASE OF VELOCITY V
MODEL.INCWATERDEPTHN	INCREASE OF WATER DEPTH AT TN

MODEL.VISCSA	VISCOSITY OF SPALART-ALLMARAS
MODEL.RESTART_PERIOD	RESTART FILE PRINTOUT PERIOD
MODEL.RESTART_RECORD	RECORD NUMBER IN RESTART FILE
MODEL.SECCURRENTS	USE OF SECONDARY CURRENTS?
MODEL.SEC_R	REVERSE OF LOCAL RADIUS
MODEL.NESTOR	USE OF NESTOR?
MODEL.ZREFLEVEL	REFERENCE LEVEL FOR NESTOR

Table 2.1: Accessible variables through the API for t2d

Variable name	Definition
MODEL.AT	CURRENT TIME
MODEL.RHO0	REFERENCE WATER DENSITY
WAQTEL.C_ATMOS	ATMOSPHERE-WATER EXCHANGE MODEL COEFFICIENT
WAQTEL.CP_EAU	WATER SPECIFIC HEAT
MODEL.DUREE	DURATION
MODEL.BCFILE	BOUNDARY CONDITION FILE NAME
MODEL.BND_TIDE	OPTION FOR TIDAL BOUNDARY CONDITIONS
MODEL.COTIMP	PRESCRIBED ELEVATION ON BOUNDARY
MODEL.VITIMP	PRESCRIBED VELOCITY ON BOUNDARY
MODEL.DEBIMP	PRESCRIBED DISCHARGE ON BOUNDARY
MODEL.QSCE	SOURCE DISCHARGE
MODEL.USCE	SOURCE VELOCITY ALONG X AXIS
MODEL.VSCE	SOURCE VELOCITY ALONG Y AXIS
MODEL.WSCE	SOURCE VELOCITY ALONG Z AXIS
MODEL.XSCE	COORDINATE X OF SOURCE
MODEL.YSCE	COORDINATE Y OF SOURCE
MODEL.ZSCE	SOURCE ELEVATION
MODEL.BETAC	BETA EXPANSION COEFFICIENT FOR TRACERS
MODEL.TRAC0	INITIAL VALUES OF TRACERS
MODEL.T0AC	REFERENCE CONCENTRATION OF TRACERS
MODEL.TRACER	VALUES OF TRACERS AT LIQUID BOUNDARIES
MODEL.TASCE	TRACER SOURCE VALUE
MODEL.DEBUG	ACTIVATING DEBUG MODE
MODEL.FLUX_BOUNDARIES	FLUX AT BOUNDARIES
MODEL.GEOMETRYFILE	NAME OF THE GEOMERY FILE
MODEL.GRAPH_PERIOD	GRAPHICAL OUTPUT PERIOD
MODEL.HBOR	BOUNDARY VALUE ON H FOR EACH BOUNDARY POINT
MODEL.IKLE	CONNECTIVITY TABLE BETWEEN ELEMENT AND NODES
MODEL.NACHB	NUMBERS OF PROC CONTAINING A GIVEN POINT
MODEL.KNOLG	GIVES THE INITIAL GLOBAL NUMBER OF A LOCAL POINT
MODEL.INCWATERDEPTH	INCREASE IN THE DEPTH OF THE WATER
MODEL.LIHBOR	BOUNDARY TYPE ON H FOR EACH BOUNDARY POINT

MODEL.LISTIN_PERIOD	LISTING OUTPUT PERIOD
MODEL.LIUBOF	TYPE OF BOUNDARY CONDITIONS FOR U ON THE BOT
MODEL.LIVBOF	TYPE OF BOUNDARY CONDITIONS FOR V ON THE BOT
MODEL.LIWBOF	TYPE OF BOUNDARY CONDITIONS FOR W ON THE BOT
MODEL.LIUBOL	TYPE OF BOUNDARY COND FOR U ON THE LAT BOUND
MODEL.LIVBOL	TYPE OF BOUNDARY COND FOR V ON THE LAT BOUND
MODEL.LIWBOL	TYPE OF BOUNDARY COND FOR W ON THE LAT BOUND
MODEL.LIUBOS	TYPE OF BOUNDARY CONDITIONS FOR U AT THE FS
MODEL.LIVBOS	TYPE OF BOUNDARY CONDITIONS FOR V AT THE FS
MODEL.LIWBOS	TYPE OF BOUNDARY CONDITIONS FOR W AT THE FS
MODEL.KP1BOR	POINTS FOLLOWING AND PRECEDING A BOUNDARY POINT
MODEL.LT	CURRENT TIME STEP
MODEL.TA	TRACER VALUE
MODEL.NBOR	GLOBAL NUMBER OF BOUNDARY POINTS
MODEL.NELEM	NUMBER OF ELEMENT IN THE MESH
MODEL.NELMAX	MAXIMUM NUMBER OF ELEMENTS ENVISAGED
MODEL.NPOIN	NUMBER OF POINT IN THE MESH
MODEL.NPOIN2	NUMBER OF POINT IN THE 2D MESH
MODEL.NPLAN	NUMBER OF PLANE IN THE 3D MESH
MODEL.NTRAC	NUMBER OF TRACERS
MODEL.MAXTRA	MAXIMUM NUMBER OF TRACERS
MODEL.MAXSCE	MAXIMUM NUMBER OF SOURCES
MODEL.NPTFR	NUMBER OF BOUNDARY POINTS
MODEL.NPTFR2	NUMBER OF BOUNDARY POINTS IN MESH2D
MODEL.BND_COLOR	BOUNDARY_COLOUR INFORMATION
MODEL.NTIMESTEPS	NUMBER OF TIME STEPS
MODEL.NUMLIQ	LIQUID BOUNDARY NUMBERS
MODEL.RESULTFILE	NAME OF THE RESULT FILE
MODEL.RESULT2D	NAME OF THE 2D RESULT FILE
MODEL.SEALEVEL	COEFFICIENT TO CALIBRATE SEA LEVEL
MODEL.PRANDTL	PRANDTL NUMBER
MODEL.TIDALRANGE	COEFFICIENT TO CALIBRATE TIDAL RANGE
MODEL.TIDALVELOCITY	COEFFICIENT TO CALIBRATE TIDAL VELOCITY
MODEL.UBOR2D	BOUNDARY VALUE ON U FOR EACH BOUNDARY POINT
MODEL.VBOR2D	BOUNDARY VALUE ON V FOR EACH BOUNDARY POINT
MODEL.UBORF	PRESCRIBED VELOCITY U ON THE BOTTOM
MODEL.VBORF	PRESCRIBED VELOCITY V ON THE BOTTOM

MODEL.WBORF	PRESCRIBED VELOCITY W ON THE BOTTOM
MODEL.UBORL	PRESCRIBED VELOCITY U ON THE LATERAL BOUNDARY
MODEL.VBORL	PRESCRIBED VELOCITY V ON THE LATERAL BOUNDARY
MODEL.WBORL	PRESCRIBED VELOCITY W ON THE LATERAL BOUNDARY
MODEL.UBORS	PRESCRIBED VELOCITY U AT THE FREE SURFACE
MODEL.VBORS	PRESCRIBED VELOCITY V AT THE FREE SURFACE
MODEL.WBORS	PRESCRIBED VELOCITY W AT THE FREE SURFACE
MODEL.VELOCITYU	VELOCITY ON U
MODEL.VELOCITYV	VELOCITY ON V
MODEL.VELOCITYW	VELOCITY ON W
MODEL.AK	TURBULENT KINETIC ENERGY K
MODEL.AKN	TURBULENT KINETIC ENERGY K AT TN
MODEL.EP	TURBULENT DISSIPATION EPS
MODEL.EPN	TURBULENT DISSIPATION EPS AT TN
MODEL.RUGOF	FRICTION COEFFICIENT
MODEL.WINDX	VELOCITY X OF THE WIND
MODEL.WINDY	VELOCITY Y OF THE WIND
WAQTEL.TAIR	AIR TEMPERATURE
MODEL.WATERDEPTH	WATER DEPTH
MODEL.X	X COORDINATES FOR EACH POINT OF THE MESH
MODEL.XNEBOR	NORMAL X TO 1D BOUNDARY POINTS
MODEL.Y	Y COORDINATES FOR EACH POINT OF THE MESH
MODEL.YNEBOR	NORMAL Y TO 1D BOUNDARY POINTS
MODEL.EQUATION	NAME OF THE EQUATION USED
MODEL.BOTTOMELEVATION	BOTTOM ELEVATION
MODEL.FLOWRATEQ	FLOW RATE
MODEL.QBOR	BOUNDARY VALUE ON Q FOR EACH BOUNDARY POINT
MODEL.EBOR	BOUNDARY VALUE ON E FOR EACH BOUNDARY POINT
MODEL.FLBOR	BOUNDARY VALUE ON ZF FOR EACH BOUNDARY POINT
MODEL.TOB	SHEAR STRESS
MODEL.CLU	BOUNDARY TYPE ON U FOR EACH BOUNDARY POINT
MODEL.CLV	BOUNDARY TYPE ON V FOR EACH BOUNDARY POINT
MODEL.LIQBOR	BOUNDARY TYPE ON Q FOR EACH BOUNDARY POINT
MODEL.LIEBOR	BOUNDARY TYPE ON E FOR EACH BOUNDARY POINT
MODEL.NSICLA	NUMBER OF SIZE-CLASSES OF BED MATERIAL
MODEL.NOMBLAY	NUMBER OF LAYER IN THE BED

MODEL.CONCENTRATION	CONCENTRATION AT TIME N
MODEL.EVOLUTION	BED EVOLUTION
MODEL.PARTHENIADES	PARTHENIADES EROSION COEFICIANT FOR EACH BED LAYER FOR EADCH CLASS
MODEL.DCLA	MEDIAN GRAIN SIZE
MODEL.SHIELDS	CRITICAL SHIELDS PARAMETER
MODEL.XWC	SETTLING VELOCITY

Table 2.2: Accessible variables through the API for t3d

Variable name	Definition
MODEL.BCFILE	BOUNDARY CONDITION FILE NAME
MODEL.DEBUG	ACTIVATING DEBUG MODE
MODEL.GEOMETRYFILE	NAME OF THE GEOMERY FILE
MODEL.IKLE	CONNECTIVITY TABLE BETWEEN ELEMENT AND NODES
MODEL.NACHB	NUMBERS OF PROC CONTAINING A GIVEN POINT
MODEL.KNOLG	GIVES THE INITIAL GLOBAL NUMBER OF A LOCAL POINT
MODEL.LIHBOR	BOUNDARY TYPE ON H FOR EACH BOUNDARY POINT
MODEL.LIUBOR	BOUNDARY TYPE ON U FOR EACH BOUNDARY POINT
MODEL.LIVBOR	BOUNDARY TYPE ON V FOR EACH BOUNDARY POINT
MODEL.NELEM	NUMBER OF ELEMENT IN THE MESH
MODEL.NELMAX	MAXIMUM NUMBER OF ELEMENTS ENVISAGED
MODEL.NPOIN	NUMBER OF POINT IN THE MESH
MODEL.NPTFR	NUMBER OF BOUNDARY POINTS
MODEL.NTIMESTEPS	NUMBER OF TIME STEPS
MODEL.RESULTFILE	NAME OF THE RESULT FILE
MODEL.X	X COORDINATES FOR EACH POINT OF THE MESH
MODEL.Y	Y COORDINATES FOR EACH POINT OF THE MESH
MODEL.EQUATION	NAME OF THE EQUATION USED
MODEL.KP1BOR	ARRAY OF NEIGHBOORS FOR EACH ELEMENT
MODEL.WAVEPHASE	PHASE OF THE WAVE

Table 2.3: Accessible variables through the API for art

Variable name	Definition
MODEL.AT	CURRENT TIME
MODEL.DT	TIME STEP
MODEL.RAISF	FREQUENTIAL RATIO
MODEL.APISET	API SET FRICTION VELOCITY
MODEL.LT	CURRENT TIME STEP
MODEL.BCFILE	BOUNDARY CONDITION FILE NAME
MODEL.DEBUG	ACTIVATING DEBUG MODE

MODEL.GEOMETRYFILE	NAME OF THE GEOMERY FILE
MODEL.IKLE	CONNECTIVITY TABLE BETWEEN ELEMENT AND NODES
MODEL.NACHB	NUMBERS OF PROC CONTAINING A GIVEN POINT
MODEL.KNOLG	GIVES THE INITIAL GLOBAL NUMBER OF A LOCAL POINT
MODEL.NELEM	NUMBER OF ELEMENT IN THE MESH
MODEL.NELMAX	MAXIMUM NUMBER OF ELEMENTS ENVISAGED
MODEL.NPOIN	NUMBER OF POINT IN THE MESH
MODEL.NPTFR	NUMBER OF BOUNDARY POINTS
MODEL.NTIMESTEPS	NUMBER OF TIME STEPS
MODEL.NFREQ	NUMBER OF DISCRETISED FREQUENCIES
MODEL.NDIRE	NUMBER OF DISCRETISED DIRECTIONS
MODEL.RESULTFILE	NAME OF THE RESULT FILE
MODEL.X	X COORDINATES FOR EACH POINT OF THE MESH
MODEL.Y	Y COORDINATES FOR EACH POINT OF THE MESH
MODEL.BOTTOM	BOTTOM
MODEL.WINDX	WINDX
MODEL.WINDY	WINDY
MODEL.HM0	HM0
MODEL.USTAR	USNEW
MODEL.FREQ	FREQ
MODEL.TETA	TETA
MODEL.FMOY	FMOY
MODEL.VARIAN	VARIAN
MODEL.TM01	TM01
MODEL.TM02	TM02
MODEL.PFREAD5	PFREAD5
MODEL.PFREAD8	PFREAD8
MODEL.VX_CTE	VX_CTE
MODEL.VY_CTE	VY_CTE
MODEL.ENERGYDENSITY	SF
MODEL.BETAM	BETAM
MODEL.ALPHA	CHARNOCK CONSTANT
MODEL.SBREK	CHOICE OF THE BREAKING MODEL
MODEL.IQBBJ	CHOICE OF THE QB COMPUTATION METHOD (BJ)
MODEL.SMOUT	CHOICE OF THE WHITE CAPPING MODEL
MODEL.CMOUT1	WHITE CAPPING DISSIPATION COEFFICIENT
MODEL.CMOUT2	WHITE CAPPING WEIGHTING COEFFICIENT
MODEL.CMOUT3	WESTHUYSEN DISSIPATION COEFFICIENT
MODEL.CMOUT4	SATURATION THRESHOLD FOR THE DISSIPATION
MODEL.CMOUT5	WESTHUYSEN WHITE CAPPING DISSIPATION
MODEL.CMOUT6	WESTHUYSEN WEIGHTING COEFFICIENT

Table 2.4: Accessible variables through the API for wac

In addition to the instance definition, the module includes all routines needed to manipulate it (creation, deletion, and so on).

2.1.3 Variable control

The way in which the instance is defined (pointers) allows manipulation of variables during the simulation. So, to get information on the variables the following set of functions has been implemented:

- `get_var_list` get the list of variables reachable with the API:
- `get_var_type` get the type of a variable.
- `get_var_size` get the size of a variable.
- `get_*`, `set_*` access to a given index of a variable.
- `get_double_array` and `set_double_array` to optimize access to a whole array instead of just a value a new pair of functions was created.

The `get_*` exist for 4 kind of types (boolean, integer, double/float, string). However the type distinction is removed in TELAPY based on Python benefits.
All of those functions are in `api_interface.f` they all take as arguments a short name (t2d, t2d, sis...) and an id.
The detailed information for all the routines are available in the Doxygen documentation.

2.1.4 Computation control

The computation control is carried out using some specific routines to launch the simulation. These routines constitute a decomposition of the main program of each TELEMAC SYSTEM modules corresponding to the following different computation steps:

- `run_set_config_t2d`: Configuration setup. This function initialises the instance and the listing output. The instance, characterised by the **id** integer parameter, represents a run of TELEMAC-2D.

  > Comment:
  > In the current version you can have only one instance of each module running at the same time.

- `run_read_case_t2d`: Reading the TELEMAC-2D steering file. This function reads the case file and set the variable of the TELEMAC-2D steering file accordingly.

  > Warning:
  > With the API we are not using the temporary folder (this folder was created by the Python environment and all the file declared in the steering file where copied and renamed inside that folder) which means that the name and path given in the steering file will be used. This also this means that the same file cannot be used for multiple keywords.

- `run_allocation`: Memory allocation. This function runs the allocation of all the data needed in TELEMAC-2D. Any modifications to quantities of TELEMAC-2D should be done before the call to that function.

- `run_init_t2d`: Initialization. This function will do the setting of the initial conditions of TELEMAC-2D. It corresponds to the time-step 0 of a TELEMAC-2D run.
- `run_timestep_t2d`: Computation function that runs one time-step of TELEMAC-2D and writing the results. To compute all time steps, a loop on this function must be done.
- `run_timestep_compute_t2d`: Computation function that runs one time-step of TELEMAC-2D without writing the results (only in TELEMAC-2D).
- `run_timestep_res_t2d`: Computation function that runs the writing part of a time step on listing and/or files (only in TELEMAC-2D).
- `run_finalize_t2d`: Finalization. This function concludes the run of TELEMAC-2D and will deallocate all the arrays and delete the instance. To start a new execution of TELEMAC-2D the function run_set_config must be run again.

For each routine defined above, the first argument is the identity number it is allowing all computation variables to be linked with the corresponding instance pointers. These routines are then called in the same order to insure a correct execution of the computation in the API main program.

2.1.5 Parallelisation

All steps associated with parallel computation must be performed by the user when he chooses to launch his calculation on several processors. In this case, after initializing the MPI environment, the user must partition the input files (geometry file, boundary conditions files, and so on) using the Fortran function "partel". Then, when the calculation is complete, it is necessary to merge each subdomains result files using the "gretel" routine of the TELEMAC SYSTEM. The MPI environment can then be closed.
All the get/set functions are using local numbering. To handle global number two tools are available:

- the function GLOBAL_TO_LOCAL_POINT returns the local numbering of a global id (0 if it is not on the partition).
- the array KNOLG returns the global number of a local point.

2.2 TELAPY Python module

It is relatively easy to use the Fortran API routines directly in Python using the “f2py” tool of the Python Numpy library. This tool will make it possible to compile Fortran code such as it is accessible and usable in Python. For more details on this tool, the interested reader can refer directly to [1]. However, using the advantage of the Python language, it is possible to implement a wrapper in order to provide user friendly function of the Fortran API. Thus, a Python overlay was developed in order to encapsulate and simplify the different API Python calls. The different Python functions written to simplify the use of API are available in the directory "$HOMETEL/scripts/python3/telapy/$".
A Doxygen documentation (In `<root>/documentation/doxypydocs` if you are on the main branch, run `doc_telemac.py -M doxypydocs` to generate it) is available and allows the user to visualize Python classes, functions that can be used as well as its input and output variables and so on.
In order to launch the Doxygen documentation, the user needs to copy and paste the link `$HOMETEL/documentation/doxypydocs/html/index.html` into his favorite internet browser. Then, the user can navigate in the Doxygen environment in order to find information.

3. Getting Started with TELAPY module

3.1 TELAPY module installation

In order to be able to use TELAPY module, the TELEMAC SYSTEM and all its external libraries must be compiled in dynamic form. The explanation of dynamic compilation is available on the TELEMAC SYSTEM website in the wiki category "installation notes" (`http://wiki.opentelemac.org/doku.php?id=installation_notes_2_beta`).
Then after compiling the module, the use of TELAPY is presented and explained in some notebooks documentation. In fact, the TELAPY module is provided with some tutorial intended for people who want to run TELEMAC-2D in an interactive mode with the help of the Python programming language.

3.2 How to run notebook documentation

Html version of the notebooks (not executable) are available in release version (starting from v8p2) in `$HOMETEL/documentation/notebooks/index.html` (Run it through a Internet browser).
In order to use notebooks, the user needs to install a notebook viewer such as jupyter notebook. Notebook documents (or "notebooks", all lower case) are documents which contain both computer code (e.g. Python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc...) as well as executable documents which can be run to perform data analysis.
First and foremost, the Jupyter Notebook is an interactive environment for writing and running code. The notebook is capable of running code in a wide range of languages. However, each notebook is associated with a single kernel. This notebook is associated with the IPython kernel, therefore runs Python code. More details on the installation and use can be found in the Jupyter website `http://jupyter.org`.

3.3 Notebook examples in TELAPY

As already mentioned, the TELAPY module is provided with some tutorial intended for people who want to run TELEMAC-2D (Or another module) in an interactive mode with the help of the Python programming language.
In order to see and run the notebook examples, the user need to launch the command "jupyter notebook $HOMETEL/notebooks/index.ipynb". This will launch a Jupyter server and a page

should appear in your default internet browser. This page in the index for all the notebooks of TELEMAC SYSTEM you can find the TELAPY ones in the TELAPY section.

4. Developer manual

This section will describe firstly how the API is implemented in Fortran. The first section will describe the instance type used in the API. Same as before we will be using TELEMAC2D in this section but it is the same behavior with the other modules. This structure shows all variables accessible by the Fortran API. Then, a section is dedicated to guidelines and advises to allow a user to add access to a new variable from TELEMAC-2D to the API.
Then the second part is focused on the Python guidelines and convention of the TELAPY module.

4.1 Instance

As already explained in section 2, in order to control the data used by TELEMAC-2D a Fortran Structure called "instance" containing all global variables of TELEMAC-2D is used. A part of this structure is presented below. In addition, some functions, described below, are available in order to handle that structure. All the instance functions can be found in the Fortran module `m_instance_t2d` (here is small part of the instance).

```
type instance_t2d
   ! run position
   integer myposition
   ! list of all the variable for model
   type(bief_obj), pointer :: hbor
!
   type(bief_mesh), pointer :: mesh
!
   type(bief_obj), pointer :: lihbor
!
   integer,         pointer :: nit
   integer,         pointer :: lt
!
   type(bief_file), pointer :: t2d_files(:)
   integer :: maxlu_t2d
   integer :: maxkey
   integer, pointer :: t2dcli
!
   character(len=144), pointer :: coupling
```

```
!
  end type ! model_t2d
```

During the use of the API, two arrays are available in order to keep track the used instances.

```
INTEGER, PARAMETER :: MAX_INSTANCES=10
TYPE(INSTANCE_T2D), POINTER :: INSTANCE_LIST(:)
LOGICAL, ALLOCATABLE :: USED_INSTANCE(:)
```

Where `INSTANCE_LIST` will contain all the instances used and `USED_INSTANCE` tells you if the id is used.

4.1.1 Instance functions

In addition to the instance definition, the API includes all routines needed to manipulate it:

- `CHECK_INSTANCE_T2D`. This function just checks that the id number is valid (between 1 and max_instances).
- `CREATE_INSTANCE_T2D`. This function creates new instance and returns the id of that instance.
- `UPDATE_INSTANCE_T2D`. This function updates the link of the instance with the variables in declarations_module.f. This is necessary as we are using pointers some are initialised later in the computation.
- `DELETE_INSTANCE_T2D`. This function deletes the instance and make the id available.

4.1.2 How to add access to a new variable

In order to add access to a new variable via the API, the following steps must be done:

1. Get the name of the variable in `declarations_telemac2d` and add ", TARGET" in its declaration.
2. Add that variable in the instance structure (file `api_instance_t2d.f`).
3. Add the initialisation in `update_instance_t2d`.
4. Add the variable in the get/set (do not forget the array ones) function that befits it.
5. Add the size of the variable in `get_var_size_t2d_d`.
6. Add the type of the variable in `get_var_type_t2d_d`.
7. Add the variable in `SET_VAR_LIST_T2D_D`.
8. Increase the value of `t2d_nb_var` in `api_handle_var_t2d.f`.

All those steps are handled through the script `scripts/add_api_variable.sh`. That takes as input a file describing the variables to add (you can see an example in `scripts/desc_file.csv`). The input file contains the following information ';' separated:

- Short name of the module: t2d, t3d, art, wac, sis...
- Api name of the variable: The name that will be given to the get/set (for example MODEL.NPOIN).
- Type of the variable:

  - We found the four from the api (`double`, `boolean`, `string` and `integer`).
  - `bief_integer` and `bief_double` when the TELEMAC-2D variable is a bief obj containing an integer/double array.
  - `bloc_double` when the TELEMAC-2D variable is a bloc (i.e. list of bief obj containing array of double).

- Variable Fortran name: Name of the variable in the `declarations_telemac2d.f`.
- Only for string contains size of the string 0 for the others.
- Number of dimension of the variable 0 for non array variable.
- readonly TRUE if the variable is cannot be set.
- get_pos: Not used yet, set to NO_POSITION.
- set_pos: Not used yet, set to NO_POSITION.
- description: Description of the variable this is the message that will be displayed by the api when looking at the list of variables.

Then to run the script just do:

```
./add_api_variable.sh desc_file.csv
```

The last thing to do is to add ", TARGET" in the declarations_*module*.f file for each of the added variable.

4.2 Coding conventions

Warning:
Be careful, the node numbering is dependent of the convention code used. When Fortran API are used the node numbering is considered from 1 to *npoin* and in python is considered from 0 to $(npoin-1)$

4.2.1 Fortran part of API

The Fortran part of the TELEMAC SYSTEM API is submitted to the main rules presented in the developer guide.

4.2.2 TELAPY module

The Python part is developed with the python convention "PEP 8". The aim of PEP 8 guidelines used in the TELAPY module is to improve the readability of code and make it consistent across the wide spectrum of Python code. A style guide is about consistency. Consistency with this style guide is important. Consistency within a project is more important. Consistency within one module or function is the most important.
The PEP 8 convention coding can be easily checked using the Pylint code analyser `https://www.pylint.org`. Pylint is a tool that checks for errors in Python code, tries to enforce a coding standard and looks for code smells. It can also look for certain type errors, it can recommend suggestions about how particular blocks can be refactored and can offer you details about the code's complexity. Pylint will display a number of messages as it analyzes the code and it can also be used for displaying some statistics about the number of warnings and errors found in different files. The messages are classified under various categories such as errors and warnings.

4.3 Validation

An example of each module/coupling should be added in `examples/python3/TelApy_api/vnv_api.py` this example should do a double run and do some get/set (see other file for example).
The option `--api` can be used in `validate_telemac.py`. This will do the following for each `vnv_*.py` :

- Every time a study (vnv_1) is added in the "pre" function it will add a new command (vnv_1_api).
- Then it will copy all the files in the study folder (vnv_1) into the api folder (vnv_1_api).
- If some files are used for multiple keywords (for example in coupling or with the RESTART FILE being used as GEOMETRY FILE as well) which is not allowed with the api a copy will be made and the steering modified accordingly.
- The command used in the api study (vnv_1_api) will be "mpirun -n x template.py module cas –double-run". This will two run of the case using the TELAPY module. We add the option double-run which will run the api twice. This is to check that variable are properly initialised.
- Finally we add a binary-wise comparison of all the output file of the study (vnv_1) versus the one of the api command (vnv_1_api).

All of those step are added this means that the normal validation will be done and api ones. This takes a while as in the end each study will be run 3 times (normal, api, api).

5. Outlooks

The work on TELEMAC SYSTEM interoperability is always in progress. In fact, all modules have not yet API structure. This will be completed in future with at least one new module available per new TELEMAC SYSTEM version. Moreover, for the existing API, some works must be done in order to obtain a full API:

- Switch the instance use, normally API modules should be pointing on the instance and not the other way around. This will allow multiple instances of considered module to run at the same time.
- Add more variables (the one from the steering file for example)
- Remove all uses of save and common in the code they can induce memory leaks.

The outlooks in longer term vision is to remove "user fortran" because all modifications can be done directly via the APIs. Also, the coupling between modules using the APIs will be rewritted. And finally, more tutorials will be added in notebook format in order to facilitate the user life with APIs.

[1] PETERSON P. F2py: a tool for connecting fortran and python programs. *International Journal of Computational Science and Engineering*, 4(4):296–305, january 2009.