

# OtFMI, an OpenTURNS module for uncertainties analysis with 0D/1D system models

Michaël Baudin <sup>1</sup>   Audrey Jardin <sup>1</sup>   Mathias Bouquerel <sup>1</sup>  
Anne-Laure Popelin <sup>1</sup>   Audrey Jardin <sup>1</sup>  
Julien Schueller <sup>2</sup>   Sylvain Girard <sup>2</sup>

<sup>1</sup>EDF R&D. 6, quai Watier, 78401, Chatou Cedex - France, [michael.baudin@edf.fr](mailto:michael.baudin@edf.fr)

<sup>2</sup>Phimeca Engineering. 18/20 boulevard de Reuilly, 75012 Paris - France,  
[girard@phimeca.com](mailto:girard@phimeca.com)

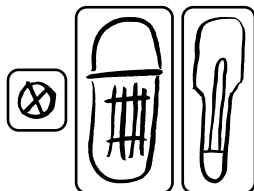
October 19th 2017

# Industrial issue

- ▶ EDF uses 0D/1D system models programmed in Modelica as decision support for the conception and operation of its industrial assets.
- ▶ How to apply OpenTURNS' panoply of methods to these models?

# “Regular” models vs 0D/1D system models

“Regular”  
modelling



“Packing”

$$\begin{cases} \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t) \\ \mathbf{y}(t) = g(\mathbf{x}(t), \mathbf{u}(t), t) \end{cases}$$

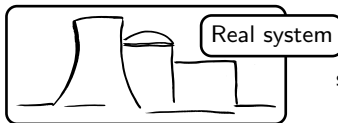
Equation formulation



Solver  
programming

# “Regular” models vs 0D/1D system models

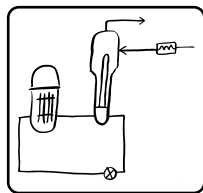
“Regular”  
modelling



0D/1D  
system modelling



“Packing”



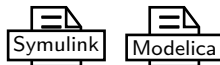
Equation formulation

$$\begin{cases} \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t) \\ \mathbf{y}(t) = g(\mathbf{x}(t), \mathbf{u}(t), t) \end{cases}$$

$$\begin{cases} \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t) \\ \mathbf{y}(t) = g(\mathbf{x}(t), \mathbf{u}(t), t) \end{cases}$$

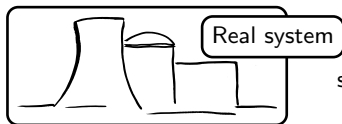


Solver  
programming



# “Regular” models vs 0D/1D system models

“Regular”  
modelling



0D/1D  
system modelling

Long (minute → hour)

CPU time

Short (second → minute)

2D, 3D

Dimensions

0D, 1D

Discrete versioning

Development rate

Continuous mutation

Numerical analyst, physicist

Actors

Design, process, operation...  
engineer

CFD, finite elements,  
C++, FORTRAN...

Tools

Modelica, Simulink...

# Modelica programming language

- ▶ Modelica is an open language for programming models based on **differential algebraic systems of equations**



- ▶ Equations are written in almost **natural language**, and solved by a multipurpose third party tool.
- ▶ It is object-oriented: available **module libraries** cover most applications
  - ▶ Complex models can be achieved simply by combining this modules using a graphical interface!

# Modelica tools

- ▶ Main tools :

- ▶ Dymola (Dassault Systèmes, proprietary)
- ▶ OpenModelica (Open Source Modelica Consortium, open source)

- ▶ Functions

- ▶ Flatten equation systems
- ▶ Compile to machine code after including a solver
- ▶ Development environment
- ▶ Model graphical interface
- ▶ Basic post-processing...

# OpenModelica, model graphical view

The screenshot displays the OpenModelica software interface. The top menu bar includes File, Edit, View, Simulation, FMI, Export, Debug, Git, Tools, and Help. Below the menu is a toolbar with various icons for file operations, simulation, and debugging.

The left sidebar shows the "Libraries Browser" with a search filter. The "Libraries" list includes OpenModelica, Modelica, and various sub-libraries like UsersGuide, Blocks, StateGraph, Electrical, Magnetic, Mechanics, Fluid, Media, Thermal, and Fluid...Flow. The "Fluid...Flow" library is expanded, showing sub-libraries like Use...ide and Examples.

The main workspace displays a model diagram titled "PumpAndValve" and "IndirectCooling". The diagram shows a complex system with multiple components, including pumps, pipes, and heat exchangers, connected by lines representing fluid flow. The components are labeled with names like "ambient", "outerPipe", "innerPipe", "pipe1", "heatFlow", and "pressure".

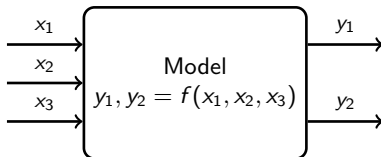
At the bottom, the "Messages Browser" shows a scripting error message: "[1] 17:09:32 Scripting Error sh: 1: impact: Permission denied".

The bottom status bar indicates the current state: Welcome, Modeling, Plotting, and Debugging.



# Piloting models

- ▶ Most OpenTURNS methods apply to functional **black boxes**
  - ▶ Uncertainty propagation and reliability analysis
  - ▶ Sensitivity analysis
  - ▶ Emulation
  - ▶ Parameter estimation



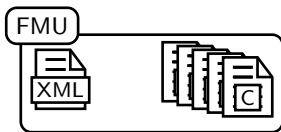
- ▶ We need efficient **input-output data interfaces**,  
a.k.a. *wrappers* in OpenTURNS jargon.

# Functional mock-up interface (FMI)

- ▶ FMI is a standard for input–output data interface for numerical model.



- ▶ A **functional mock-up unit (FMU)** is a black box following the standard.

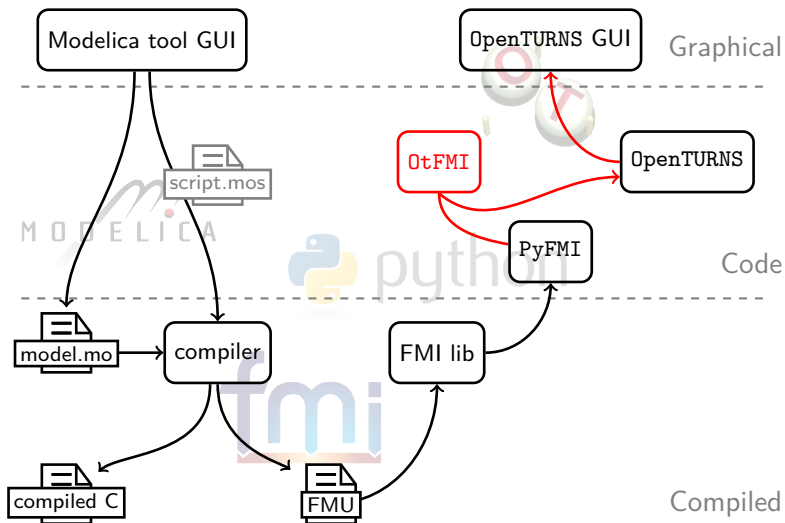


# OtFMI: integrating FMI support into OpenTURNS

- ▶ The new open source module OtFMI allows transparent use of FMU with OpenTURNS methods.
- ▶ It provides high level classes derived from `ot.PythonFunction`: running an FMU instead of a Python function only requires to **change a single code line!**

```
import otfmi
otfmi.FMUFunction("path/to/fmu",
                  inputs_fmu=["x_1", "x_2", "x_3"],
                  outputs_fmu=["y"])
```

# Implementation overview



# OtFMI graphical interface

## Motivations

- ▶ Provide access to OpenTURNS' methods for Modelica users unfamiliar with Python
- ▶ Considerably ease simple studies

## Issues

- ▶ Modelica models often define **hundreds or thousands of variables**

# OtFMI graphical interface, FMU overview

OTGui - [Modèle physique FMI]

Fichier Vue

**Etudes**

- OTStudy\_0
  - Modèles physiques**
    - PhysicalModel\_0
      - Définition**
        - Modèle probabiliste

Propriétés Variables Différentiation

FMU file

Identifiant	bil100partiel_GV_0Unit2_0wlnit
Version FMI	2.0
Outil	Dymola Version 2017 (32-bit), 2016-04-12 (using dassl with tolerance 0.0001)
Plateforme	default
Auteur	
Version	
Copyright	
Date/Heure	2016-07-12T11:49:23Z
GUID	{44d61dd0-e7a1-4166-ada8-ecdc48a166b7}
Nombre de variables	2451
Causalité	paramètre : 119, entrée : 0, sortie : 0, locale : 2332

# OtFMI graphical interface, picking inputs and outputs

The screenshot shows the OtGui - [Modèle physique FMI] window. The 'Variables' tab is selected. The left sidebar shows a tree view with 'OTStudy\_0' expanded, containing 'Modèles physiques' and 'PhysicalModel\_0'. The 'PhysicalModel\_0' node is expanded, showing 'Définition' and 'Modèle probabiliste'. The main panel has a 'Filtres' section with a search bar and dropdowns for 'Variabilité', 'Causalité', and 'E/S'. Below this is a 'Variables' list box containing the following variables: rho, GV1\_CFD1, GV2\_CFD1, GV3\_CFD1, GV4\_CFD1, VVP\_1A\_VAPF, VVP\_2A\_VAPF, VVP\_3A\_VAPF, VVP\_4A\_VAPF, and DB\_GV1\_CFD1. At the bottom, a table displays the selected variables with their descriptions, variability, causality, E/S type, and values.

Nom	Description	Variabilité	Causalité	E/S	Valeur
QARE0		Fixe	Paramètre	Entrée	2126,272222
PARE0		Fixe	Paramètre	Entrée	7032780
TARE0		Fixe	Paramètre	Entrée	499,9305
PGCT0		Fixe	Paramètre	Entrée	6540000
QGS0		Fixe	Paramètre	Entrée	183,4088889
OGRE0		Fixe	Paramètre	Entrée	1922,635556
BIL100...		Fixe	Locale	Sortie	3788204475

Variables sélectionnées : entrée : 6, sortie : 1

Buttons: Fichier, Vue, Propriétés, Variables, Différentiation, Effacer, Evaluer

# Perspectives

- ▶ Most 0D/1D system model are dynamical.  
We need methods for sensitivity analysis and emulation of model with **time series inputs or outputs**.
- ▶ EDF is interested into **data assimilation** with its Modelica models.
- ▶ What are the opportunities of **extending the Modelica language** to support stochastic description of variables?



Thank you for your attention.