DACCOSIM 2017

Jean-Philippe Tavella, Mathieu Caujolle (EDF Lab, Paris-Saclay & RISEGrid, France)
Stéphane Vialle, Virginie Galtier, Michel Ianotto, Cherifa Dad
(CentraleSupélec & RISEGrid, Université Paris-Saclay, France)
José-Juan Hernandez, José Evora (SIANI, Las Palmas, Spain)

A powerful co-simulator for Smart Power Grids

Last update: April 30th, 2017

Modelica’17: Vendor Session for Daccosim 2017
What is RISEGrid?

A French research institute dedicated to the Smart Grids

- Launched by CentraleSupélec and Electricité de France in 2012 to challenge evolutions of energetic systems
- About 20 people (academic & industrial researchers, PhDs, post-docs)
- For the study of Smart Power Grids (both TSOs & DSOs)
A French research institute dedicated to the Smart Grids

- Launched by CentraleSupélec and Electricité de France in 2012 to challenge evolutions of energetic systems
- About 20 people (academic & industrial researchers, PhDs, post-docs)
- For the study of Smart Power Grids (both TSOs & DSOs)


Some of the research topics with PhDs in progress

- Technical / economical optimization of power systems with renewable energy production units
- Optimization of the grid operation with renewable energy and storage units
- Stability of the distribution grids from a control point of view
- Harmonics contribution of an installation connected to the grid
- Modeling & simulation of large-scale Smart Grids

starting the Daccosim project in 2012
Modeling CPSs

- All the physics with Modelica
- Models built with components from dedicated business libraries internally developed by EDF:
  - ThermoSysPro for thermo-hydraulic
  - BuildSysPro for energy in buildings
  - PlantSysPro for industrial processes like hot water system
  - GridSysPro for power grids
- Control parts either in Modelica or using other languages (e.g. IEC 61131-3, UML/SysML)

Simulating CPSs

- The 2.0 version of the FMI-CS standard
  - IP protection for modeled components
- Daccosim as FMI-CS co-simulator platform
  - Internal development for an agile roadmap
  - Open source code (not a commercial solution)

Solution advocated by RISEGrid

https://daccosim.foundry.supelec.fr
Key computational features:

- Runnable locally on a standard PC or on a HPC cluster
- Generalized co-initialization method
- Multiple simulation strategies:
  - Overlapped / ordered orchestration mode
  - Constant or adaptive step size (Euler, Adams Bashforth, …)
  - With or without a co-initialization step
- Event handling
  - Bisectional event detection on Booleans and integers variables (approx. event time)
  - POC\(^1\) of hybrid co-simulation (exact event detection) with hybrid FMUs exported from a development version of Dymola 2018

\(^1\) Refer to a proposal pushed by EDF in the MAP FMI this year and titled “Hybrid Co-simulation with the FMI-CS Standard”
Co-initialization method

Automatic conversion of a generalized causality graph into an acyclic one

- Automatic global dependencies calculation from internal dependencies (modelDescription file) and external dependencies (calculation schema analysis)
- Isolation of cyclic parts as SCCs (strongly coupled components) and generation of an equivalent uncyclic graph
- Newton-Raphson algorithm to solve SCCs in a parallel way
Focus on orchestration modes

Common step size negotiation

- All FMUs share the same step size
- The scheduling is controlled by a hierarchy of masters (one global, several local)
- Two data synchronization methods: prudent & optimistic modes

Relaxed synchronization (prudent mode)  Overlapped orchestration (optimistic mode)
Direct horizontal data communication (no bottleneck)

- Each FMU is embedded in a wrapper
- **FMU-wrapper** developed by [SIANI](https://bitbucket.org/siani/javafmi)
- FMU-to-FMU data exchange based on the rich 0MQ messaging library
- Asynchronous communication in parallel with computation
- FMUs can be on the same machine or on separate cluster nodes
- MPI not adapted to Daccosim
A DSL to manipulate thousands of FMUs and thousands of FMU-to-FMU data exchanged
Both directions: text-to-graphic or graphic-to-text conversion
All Daccosim commands are scriptable

Complex calculation schema automatic generation

Daccosim Script Language

Modelica’17: Vendor Session for Daccosim 2017
Matryoshka

“Russian doll” smart mechanism to export super FMUs

- Based on FMU-builder developed by SIANI
  https://bitbucket.org/siani/javafmi
- Transform a co-simulation graph with external input / output connectors into an FMU
- Automatic computation of internal dependencies
- Smart process to generate multi-threaded FMUs with embedded inner step-size control mechanism

![Diagram of Matryoshka model]

- e.g. a 4-FMU Daccosim calculation graph automatically transformed into a standard FMU
Daccosim 2017 GUI

All actions available through a one-click button toolbar

- Execute a DSL script / export a calculation graph into a text form
- Load / unload FMUs into the palette
- Define & configure external variables & input / output connectors
- blocks / cluster nodes association
- Parameterize the co-simulation
- Check a calculation schema
- Build its dependency graph
- Generate a co-simulator master code and compile it
- Execute one-node co-simulation / stop a co-simulation
- Configure real time or post-mortem visualization
- Build a Matryoshka
- Help
- …

A complete Daccosim user’s guide with some examples can be downloaded at https://daccosim.foundry.supelec.fr/pages/Documentations.html
DacRun addin for Daccosim

A CentraleSupélec solution to power Daccosim on clusters

Use Daccosim to generate one (global or local) Java code per Vnode (virtual node)

Allocate Pnodes (cluster nodes) thanks to a Linux ressource manager on the cluster

Run dacrun.py (using Parallel Python module) for:
- Automatic recovery of Pnodes from the ressource manager
- Mapping of Vnodes on Pnodes
- On each Pnode:
  - compilation (javac) of the dedicated master codes
  - run of the Java master code (one JVM per Pnode)
  - Recovery of local result files
- Not yet a global result files aggregation

DacRun: batch or interactive mode on cluster or standalone PC

A DacRun user’s guide can be downloaded at https://daccosim.foundry.supelec.fr/pages/Documentations.html
Matryoshka more efficient than a pure Dymola simulation

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean RMSE over the 12 buildings</th>
<th>Computation time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On current (A)</td>
<td>On voltage (V)</td>
</tr>
<tr>
<td></td>
<td>1 Day</td>
<td>5 Days</td>
</tr>
<tr>
<td>Dymola ($10^{-4}$ tolerance)</td>
<td>1.34</td>
<td>$8.69 \times 10^{-1}$</td>
</tr>
<tr>
<td>Dymola ($10^{-5}$ tolerance)</td>
<td>$3.25 \times 10^{-1}$</td>
<td>$2.52 \times 10^{-1}$</td>
</tr>
<tr>
<td>Dymola ($10^{-6}$ tolerance)</td>
<td>reference</td>
<td>reference</td>
</tr>
<tr>
<td>Matryoshka</td>
<td>$7.99 \times 10^{-2}$</td>
<td>$1.39 \times 10^{-1}$</td>
</tr>
</tbody>
</table>

V. Galtier, M. Ianotto, M. Caujolle, R. Corniglion, J-Ph. Tavella, J. Évora Gómez, J-J. Hernández Cabrera, V. Reinbold, E. Kremers, “Building Parallel FMUs (or Matryoshka Co-Simulations)”

In the Modelica Conference (Modelica 2017). Prague, May, 15-17, 2017
Significant size-up

- Demonstrated scalability on a system mixing over 1000 buildings connected to dozens of LV feeders, each modelled by an FMU
- Daccosim execution on a cluster with Infiniband communication network
- No significant loss of accuracy

Daccosim is developed in Java under Eclipse for both Windows and Linux on 32 or 64-bit platforms

- The code is distributed under the AGPL open source license and can be downloaded with documentation, publications and examples

- Daccosim is being improved, tested and maintained by the RISEGrid institute & SIANI

- One version per year, last one being Daccosim 2017 at the beginning of 2017
1\textsuperscript{st} demo: co-initialization

\[
\begin{align*}
(x_2 - 2) \cdot \text{time} + x_1 + x_2 &= 10 \\
2x_1 - (x_2 - \sqrt{\text{time}}) &= 11
\end{align*}
\]

A mathematical system

when time = 0 :
\[
\begin{align*}
x_1 + x_2 &= 10 \\
2x_1 - x_2 &= 11
\end{align*}
\]
\[\Rightarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 7 \\ 3 \end{pmatrix}\]

when time \to \infty :
\[
\begin{align*}
x_1 &\to -\infty \\
x_2 &\to 2
\end{align*}
\]
A queue of **barrels** waits on a conveyor to be filled. Only one barrel is filled at a time: when the water reaches a given level in the barrel, the filling process immediately switch to the next empty barrel;

A **tank** stores the water to fill the barrels and the flow rate of water filling the barrel decreases with the level of water in the tank;

A first controller **c1** manages the opening of the valve of the tank;

A second controller **c2** can abort the current barrel filling and the process must switch to another barrel without any delay. It can also change the target level of the barrels: in this case, when the command is received, the factory takes into account the new target level only after the current barrel has reached the previous target level.
when time \geq 0.5 \text{ then}
  \text{Valve} = \text{true}
elsewhen time \geq 3.2 \text{ then}
  \text{Valve} = \text{false}
elsewhen time \geq 5.2 \text{ then}
  \text{Valve} = \text{true}
end when

\[
\frac{d \text{ Water}}{dt} = \begin{cases} 
\text{inFlow} & \text{if sens} \\
-10000 & \text{else}
\end{cases}
\]

when \{ \text{Water} \geq \text{Level, Abort} \} \text{ then}
  \text{sens} = \text{false}
elsewhen \text{Water} \leq 0 \text{ then}
  \text{sens} = \text{true}
end when
Questions