Simulation of Large Grids in OpenModelica: reflections and perspectives

Francesco Casella[1] Alberto Leva[1] Andrea Bartolini[2]

[1]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy,
{francesco.casella,alberto.leva}@polimi.it
[2]Dynamica s.r.l., Italy, andrea.bartolini@dynamica-it.com

Abstract

This paper belongs to a long-term research activity on modelling and simulation of large-size power grids in Modelica, using the OpenModelica Compiler. We describe the present state of the research, its evolution over the last year, the conclusions we could reach in this period in comparison with the initial hypotheses, and some results. Finally, we outline the future of the presented activity.

Keywords: Grid Modelling and Simulation, Large-Scale Systems, Efficient Simulation.

1 Introduction

The modelling and simulation of large power grids is an emerging domain of interest for the Modelica language, as the encountered problems basically consist of large networked systems with decentralized control, where multiple producers and consumers cooperate to the goals of stable network behaviour, satisfaction of all the load requests, and system optimality.

Although control strategies for such large-scale systems are usually designed as hierarchical systems, abstracting low-level behaviours within higher levels, it is sometimes necessary to simulate the entire system. This can be the case when a full verification of the designed strategy, including the interactions among its parts, is in order—and this is an issue shared by any large-scale system.

In the case of electric grids, there is another problem to address. For management reasons at the nation- or continent-wide scale, it is required to periodically assemble a model of the entire system and use it to run numerous simulations, to verify that the stress expected in the next time period can be sustained without incurring in stability problems, to test critical manoeuvres when required, and possibly to take decisions in a view to optimise the operation. This particular use of simulation makes a fast code generation vital.

Over the last two years, we have been working on this subject, with the goal of providing an entirely Modelica-based solution using the open-source OpenModelica Compiler (OMC) for code generation. The problem at hand is one very interesting case of an emerging class of large-scale models, see (Casella, 2015) for an overall discussion on this topic. Preliminary results were presented in (Casella et al., 2016), which was mainly addressed to the power system community. This paper incorporates the results of additional work carried out since then, and presents the current state of the research from the perspective of the Modelica community.

2 Previous research

In this section we summarise the research context and the results from which we started, referring the interested reader to (Casella et al., 2016) for further details.

National grids in Europe are rapidly evolving (ENTSO-E, 2015, 2014). The penetration of intermittent sources like wind and solar enhances the need for continent-level integration for countries to help one another. Transmission networks are moving from the traditional structure dominated by large synchronous generators and AC links, toward an increasing share of HVDC links and of medium- and small-scale generators interfaced to the grid via AC/DC/AC links. As a consequence, the management of transmission grids by national Transmission System Operators (TSOs) increasingly requires knowledge of the dynamic behaviour of the the system outside the country boundaries.

Traditionally, well-established domain-specific tools are used such as PowerFactory, PSS/E, and Eurostag. These tools come with extensive component libraries, but the exact formulation of the said models is difficult to access, since they are written in low-level languages like FORTRAN. With commercial tools, the models' source code might even be unavailable to the end user. This hinders the required interoperability, as models of the same object in different tools may behave differently. Indeed, full interoperability would ideally require all European TSOs to use the same simulation tool.

Modelica has been already used for the modelling of electrical power systems, including detailed machine models (Franke and Wiesmann, 2014; Kral and Haumer, 2005), and more recently it has been considered also to model electro-mechanical transients in high-voltage generation and transmission system. In this context, an activity worth mentioning is the iTesla European FP7 research project (Vanfretti et al., 2013, 2014; Zhang et al., 2015), although the results of the project refer to small- or medium-sized power systems, with at most a few dozens generators and transmission lines.

At the beginning of this activity, we formulated the following research question: "Are Modelica and Modelica tools adequate to support the simulation of electromechanical models of national- and continental-size power grids?". From that moment till now, we have been building a prototype model library, using which many test cases have been created and analysed to answer the research question stated above. The library contains representative models for the main components used in the addressed systems, i.e., generators, governors, transformers, transmission lines, and loads. Note that the goal of this library is not the accurate modelling of any real system, but rather to build realistic models of large-scale power systems in order to test the ability of Modelica tools to handle them. The simulation code is generated with the open-source OpenModelica Compiler (OMC).

The results obtained so far are encouraging, but at the same time the activity has revealed several shortcomings of the OpenModelica environment, in particular referring to the efficiency of both the code generation and the simulation phase. A development activity was therefore carried out – and is still ongoing – within the OpenModelica Consortium to address the evidenced problems, and verify the effects of the introduced improvements with respect to some representative benchmark cases. The result of the activities just sketched is presented in the following.

3 Current research activity

For the purpose of this study, a prototype library has been built, providing models of synchronous generators, transformers, transmission lines with breakers and over-current protections, electrical loads, and governors. All the high-level modelling features of Modelica, like the support for complex numbers, were extensively used.

```
operator record ComplexVoltage = Complex(
  redeclare SI.Voltage re, redeclare SI.Voltage im);
operator record ComplexCurrent = Complex(
  redeclare SI.Current re, redeclare SI.Current im);
connector Pin
  Types.ComplexVoltage V "Line-to neutral voltage";
  flow Types.ComplexCurrent I "Line current";
end Pin;
```

Figure 1. Connector definition.

Figure 1 shows the types for complex current and voltage, used to define the electrical connector. It is assumed that the three-phase voltages and currents are always balanced and described by phasors referred to a common reference frame rotating with a reference speed/frequency, usually that of a strong generator in the network.

Under these assumptions, a three-phase voltage and current system can be described by just one voltage and one current phasor, provided the appropriate factors of 3 or $\sqrt{3}$ are taken into account when computing the actual power flows. Most large-scales grid studies are made under this assumption; extensions to unbalanced three-phase systems are feasible, but are far more computationally demanding, and outside the scope of our study.

It is also assumed that the network frequency stays close enough to its reference value, so that the impedances can be computed with that value, ans considered constant.

There are some similarities between the design of this library and that of the `Modelica.Electrical.QuasiStationary` library. However, the specific modelling framework which is required for large power grid studies, i.e., three-phase balanced systems represented by one equivalent phase only, is not directly available there.

```
algorithm
// Detection of high current - side a
  when I_a_mod > Ilmax_mod then
    TimerOn_a := true;
    TimerStartValue_a := time;
  end when;
  when I_a_mod < Ilmax_mod and pre(TimerOn_a) then
    TimerOn_a := false;
  end when;
algorithm
// Handles the actual status of the breaker - side a
  when pre(TimerOn_a) and
       time > pre(TimerStartValue_a)+Ilmax_delay then
    BreakerStatus_a := 0;
  end when;
equation
  Yl_act = Yl * Complex(BreakerStatus_a * BreakerStatus_b);
  Ysa_act = Ys * Complex(BreakerStatus_a);
  Ysb_act = Ys * Complex(BreakerStatus_b);
  Ia = Il + Isa;
  Il + Ib = Isb;
  Isa = Ysa_act * Va;
  Isb = Ysb_act * Vb;
  Il = Yl_act * Vl;
  Va = Vl + Vb;
```

Figure 2. Model of a transmission line (excerpt).

Figure 2 shows an excerpt of transmission line model, including breakers for current protection. The two algorithms compute the state of the breaker on the one side of the line (the other is omitted for brevity), while the equations describe admittances, currents and voltages.

```
equation
// ideal transformer:
  VT1 = n * Vb;
  IT1 = -Ib / n;
// actual admittances
  Yl_act = Yl * Complex(LineBreakerClosed);
  Ys_act = Ys * Complex(LineBreakerClosed);
// pi-model
  Ia = Il + Isa;
  Il = IT1 + IsT1;
  Isa = Ys_act * Va;
  IsT1 = Ys_act * VT1;
  Il = Yl_act * Vl;
  Va = Vl + VT1;
```

Figure 3. Model of a transformer (excerpt).

The equations for the transformer model are analogous—see Figure 3, where again just an excerpt is reported for brevity.

The model of a synchronous generation unit is built hierarchically, by connecting those of synchronous machine, governor, and exciter controller, see Figure 4. The synchronous generator is described by the simplest possible 4-state model, taking the mechanical power input `Pm_req` from the governor, and the normalised excitation voltage `vf` from the voltage controller.

DOI 10.3384/ecp17132227

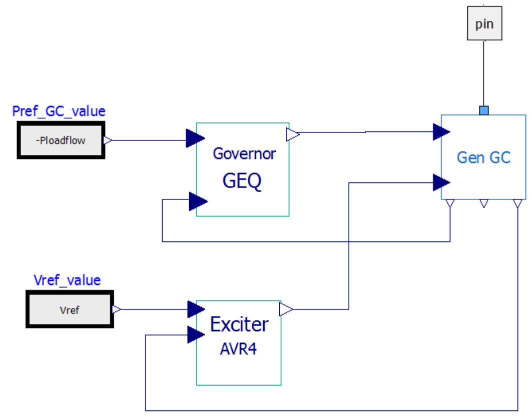

Figure 4. Generator model.

```
equation
// mechanical equations
  Snom_GC_mod*Ta*der(omega)/omega = Pm_req - Pe;
  der(delta) = omega - omega_ref;
// lead-lag vd
  Tqo*der(ed) + ed = (Xq - X)*iq;
  ed = vd - X * iq;
//lead-lag  + lag vq
  Tdo*der(eq) + eq = vf - (Xd - X)*id;
  eq = vq + X * id;
//normalization
  Vd = vd*V_ll_nom_mod;
  Vq = vq*V_ll_nom_mod;
  Id = id*Snom_GC_mod/V_ll_nom_mod;
  Iq = iq*Snom_GC_mod/V_ll_nom_mod;
// conversion from Park ref. to pin ref.
  Vpr_ll =  Vd*sin(delta) + Vq*cos(delta);
  Vpi_ll = -Vd*cos(delta) + Vq*sin(delta);
  Ipr =  Id*sin(delta) + Iq*cos(delta);
  Ipi = -Id*cos(delta) + Iq*sin(delta);
// power calculation
  Pe = 3 * (Vpr*Ipr + Vpi*Ipi);
  Qe = 3 * (Ipr*Vpi - Vpr*Ipi);
```

Figure 5. Synchronous generator equations (excerpt).

This model is interfaced to the rest of the system through a `Pin` connector (see Figure 1). The core equations are shown in the excerpt of Figure 5.

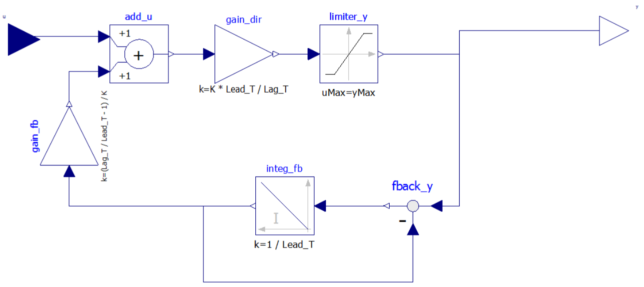

Figure 6. Excitation system model, according to IEEE Std 421.5-2005.

The governor and exciter models (see for example Figure 6) are simple block diagrams, in accordance to the IEEE standards. A graphical representation is here preferred to a text-based one, as it is immediately familiar to any practitioner in the field.

Coming to loads, both linear and nonlinear models are provided. The basic linear load model is described by the equation $V = ZI$, where Z is a constant complex impedance. PQ models can be easily obtained by writing equations that prescribe the real and imaginary parts of the complex power flow through the `Pin` connector of the load. However, doing so makes the (large) implicit system of equations describing the network nonlinear.

Since reliable sparse nonlinear solvers were not available in the Modelica tool used for this study at the beginning of the work, a linearized PQ model was also implemented, in which the relationships between complex voltage, current and power were linearized around the nominal operating point, which is supplied by an external power flow computation. Later on, as full nonlinear sparse solvers became available both for initialization and simulation, the regular PQ load models were used.

In order to simulate network protection strategies, it is necessary to be able to simulate the dynamic formation of more than one electrical islands from an single synchronous network, due to the opening of strategically placed circuit breakers. The newly formed islands need separate frequency references and may drift apart from each other. In this case, three factors shall be considered:

1. *topological factor,* i.e., detecting the formation of sub-islands in the network, starting from the actual status of circuit breakers,

2. *functional factor,* i.e., assessing the ability of each island to survive in terms of voltage and frequency regulation,

3. *modelling factor,* i.e., finding a model structure which allows the models to properly work after the islanding event in each of the possible functional condition, avoiding singularities or other numerical problems that would cause the simulation to abort.

Up to the authors' understanding, this is a major departure from the modelling assumption and the structure of all the existing Modelica libraries for multi-phase power system modelling, which assume a fixed connection topology throughout the simulation, and exploit this property to use the over-constrained connector features originally introduced in Modelica 3.0, propagating the phase reference through the connectors. Unfortunately, this feature cannot be used for the grid models considered in this paper, unless it is extended to handle dynamically changing connection graphs; this in turn would require a change of the Modelica language, and major changes to how this feature is handled in the Modelica tool.

In this study a prototype framework to manage this aspect was implemented using Modelica 3.3. In fact, for the purposes of the testing activities carried out so far, the topological analysis was not handled with a general-purpose algorithm (that could be implemented as an external C function), but rather hard-coded in simple Modelica functions that returned the results of the analysis, which were known a-priori for those tests.

In a nutshell, the prototype framework is based on a Network Supervisor model, which is unique for the entire grid model. The supervisor:

- receives the status of the network breakers via input/output connections and monitors their changes;
- performs the topological analysis and detects the formation of islands in the grid each time the breaker status changes;
- sends to each load via input/output connections the new activation status (active/not active) and to each generator the new frequency reference (or reference generator), when the breaker opening actually leads to island formation.

Generators and loads change their active equations (using conditional equations) and frequency reference, according to the information received from the Network Supervisor, in order to avoid singularities that may prevent the simulation from continuing. For example, all PQ load models are turned into open circuits when they find themselves in a not active island,, i.e., an island without generators, because otherwise the system of equations of the sub-island would have no solution, aborting the simulation.

Network	Nodes	Gens	Lines	Trafos	Equations
GRID_C	751	74	369	583	56386
GRID_E	1817	267	1458	1202	157022
GRID_D	8376	2317	1946	2489	579470
GRID_G	8113	407	6833	2824	593886

Table 1. Features of the exemplary grids.

Coming to the test cases, four exemplary grids of different sizes were considered, named in the following `GRID_{C,E,D,G}`. Table 1 summarizes the main features of the models, which describe the Irish power system, the 400 kV Italian power system, the 400 kV paneuropean transmission system, and the detailed 400-220-150-132 kV transmission system, respectively. The models were supplied by CESI in the context of the study reported in (Casella et al., 2016). Note that the number of nodes, reported for convenience, is not always a reliable complexity indicator, because a node can have a very variable number of attached entities, each in turn of different complexity; for this reason, we also report the number of equations. The results obtained by simulating these models are summarised and discussed in the next section.

4 Simulation results

During the first round of activity, that took place between November 2015 and July 2016, the only fully reliable large-scale sparse solver made available by the OpenModelica tool was the KLU linear solver, which is geared specifically towards the efficient solution of electrical circuit equations. This restricted the choice of system models to those in which the very large strong component of the causalized system equations is linear. This sub-set of equations comprises the transformer and transmission lines components (which are linear) and the load models, which can then be either constant impedances or PQ load models linearized around the nominal operating point.

The only viable integration strategy given this limitation was then to causalize the system of differential-algebraic equations, bringing it into state-space form, and then integrating it with an explicit ODE solver. At each time step, the calculation of the derivatives requires the solution of the very large strong component of the system, which is performed by the KLU sparse solver.

Steady-state initialization was also feasible by prescribing the currents at the boundaries of the synchronous generators to the values obtained by the external power-flow computations, which allows to split the initialization problem into one very large linear system (transformers + transmission lines + loads) and many small nonlinear problems (each individual synchronous generator). The availability of an external power-flow computation is also essential to set proper initial guess values on the nonlinear problems.

The models were simulated for 20 seconds, which is the typical length of transients for stability studies, using Heun's algorithm (2nd order Runge-Kutta) and a fixed time step of 20 ms. The transmission lines currents are monitored on both sides, but no breaker ever tripped.

Code generation and simulations reported here were carried out on an Intel Xeon CPU E5-2650 server with 20 virtual cores at 2.30GHz, 72 GB of RAM installed, running Linux Ubuntu 16.04 LTR 64 bit and using OMC 1.11.0-dev-59. Each simulation was carried out as a single thread, which is reasonable as multi-core systems can be exploited by running several simulation *scenarios* in parallel. The parts of the code generation process that can run independently are instead parallelised in OMC, as well as the compilation of the C code.

Network	Flattening	C gen.	Compilation	Simulation
GRID_C	24	24	13	12
GRID_E	73	67	35	44
GRID_D	334	315	123	111
GRID_G	318	303	144	186

Table 2. Performance results (times in seconds).

Performance results are summarised in Tab. 2. Notice for clarity that the third column includes both the time for structural analysis and optimisation, and that for C code generation. The fourth column is the time used by the C compiler and by the linker, while the fifth shows the total simulation time. The simulation time is almost twice as fast as real time for the smallest grid, and about 10 times slower for the largest one.

DOI 10.3384/ecp17132227

The time spent for flattening, structural analysis, C-code generation and compilation currently dominates, taking up to about 13 minutes for the largest case. This is already a feasible situation for off-line applications, in particular if one generates the simulation code once and then runs many simulations with it, by only changing the parameters in the initialization files, which can include for example the tripping times for circuit breakers, the load values, and so forth. However, such a code generation and compilation time is still definitely too long for real-time applications, with the typical turnover time of TSO operations, which is around 15 minutes. The peak recorded memory allocation was about 20 GB of RAM, which does not pose any problem on reasonably sized systems.

As to event handling, the event detection logic currently implemented in OMC uses a simple bisection algorithm to determine the exact point in time when thresholds are crossed. If a great precision is not necessary, only a few iterations would be required, whose cost will be comparable to that of carrying out a two-stage time integration step. Otherwise, it could be possible to implement a more sophisticated event detection, for example using a Newton-based algorithm.

Later on, as the sparse nonlinear solver Kinsol and the sparse DAE solver IDA became available in the OpenModelica tool, it was possible to use the full nonlinear PQ load models, as well as to employ a variable step size sparse implicit DAE solver, which turns out to be more efficient than explicit solvers as the underlying system is somewhat stiff. Note that in this case the system is not causalized and brought to state-space form; after alias reduction (and possibly index reduction, which however is not required for these specific models), the resulting DAEs are passed directly to the solver.

An example is shown in Figures 7 and 8, where three solvers are compared:

- Runge-Kutta/KLU on the grid model with linearized PQ loads,
- IDA/Kinsol/KLU on the grid model with linearized PQ loads,
- IDA/Kinsol/KLU on the grid model with nonlinear PQ loads.

The simulated transient is a 30% step reduction of the active power of one of the PQ loads (node N_152) in the smaller GRID_C model. The transients obtained with KLU and IDA/Kinsol on the linear network model match within the relative tolerance of the variable-step integrator, i.e., 10^{-6}.

Figures 7 and 8 show the frequency transient in node N_152 (load) and node N_144 (generator). The frequency peak at node N_152 is about 50.1 Hz. The blue and green (overlapped) traces refer to the PQ linearised model, integrated using the KLU and the IDA solvers respectively,

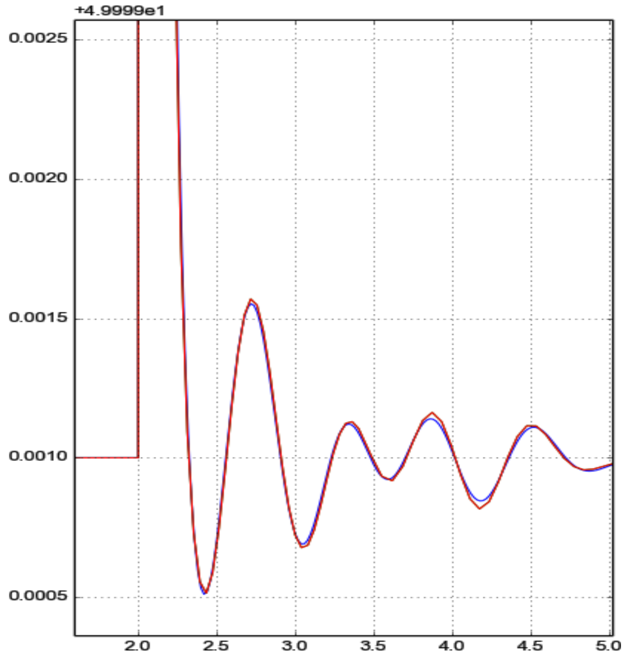

Figure 7. KLU and IDA/Kinsol test – frequency transient at load N_152.

while the red one refers to the PQ non-linear model, integrated using the IDA solver. It is apparent how the linearized model is perfectly adequate to solve this kind of transients, although it could end up being badly off in other more severe transients.

Performance results obtained with the IDA solver are reported in Table 3, using the same hardware of earlier experiments and OMC 1.12.0-dev-731. The simulation time shown is net of the time for set-up, initialisation and writing results to mass storage. Comparing these results with those of Table 2, it is apparent how this solution strategy is much more efficient, despite the additional computational complexity brought in by the nonlinear load models.

The advantage of using the variable step-size DAE solver are even more evident if longer simulation intervals are taken, as is for example the case when addressing voltage stability studies. The ability of the implicit DAE solver to take steps with a length of many seconds when the system is close to steady-state, allow to massively outperform the explicit ODE solver, whose step length is unconditionally limited to a few tens of milliseconds owing to numerical stability problems.

The times for code generation and compilation are not reported here, as these phases have not yet been optimized for this kind of solver, so that the results are not indicative of the performance that could be achieved once all current performance bottlenecks have been resolved.

Finally, the Network Supervisor prototype was tested on the same `GRID_C` model, using PQ linearized load models, and changing some over-current protection thresholds in such a way to result in the opening of four lines after 1 s from the simulation start. These lines opening generate three sub-islands.

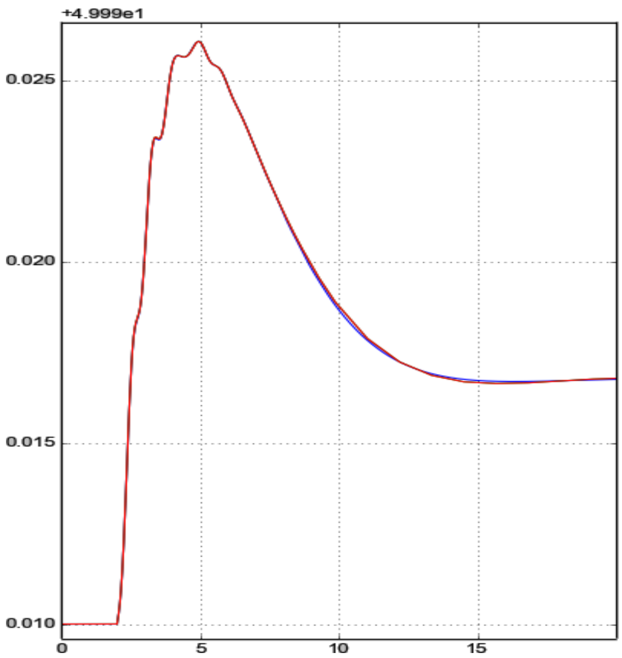

Figure 8. KLU and IDA/Kinsol test – frequency transient at generator N_144.

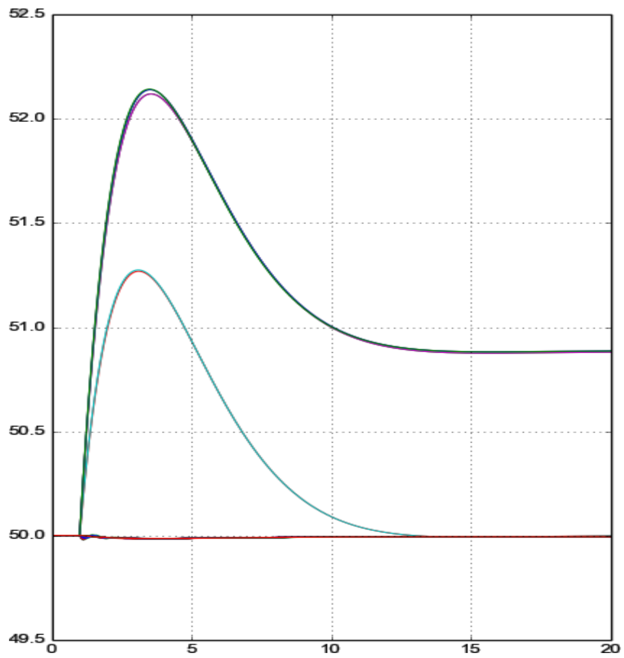

Figure 9. Network generator frequencies.

Table 3. Simulation performance with the IDA sparse DAE solver

Network	Rel. tol.	No. of steps	Sim. time [s]
GRID_C	10^{-4}	39	0.96
GRID_C	10^{-6}	146	3.18
GRID_E	10^{-4}	140	8.80
GRID_E	10^{-6}	364	15.22
GRID_G	10^{-4}	221	59.95
GRID_G	10^{-6}	615	123.19

- *Sub-island 1,* which contains only three generators, two of these in frequency regulation. The generators will be shut down, bringing their power output to zero rapidly.

- *Sub-island 2,* a small sub-island with six generators. All generators are kept in regulation and a new reference generator will be assigned (N_517).

- *Sub-island 3,* a big sub-island, which contains the rest of the network. All generators are kept in regulation and the reference generator does not change.

Figure 9 shows the new frequency rearrangement after the protection opening. Starting from the top, the first trace refers to the sub-island 2, which reaches a new steady-state with a frequency deviation of about 0.9 Hz; the second trace refers to the sub-island 1, which is shut down, and shows a transient with a frequency peak deviation of about 1.2 Hz; the last trace refers to the sub-island 3, which is the most stable due to its large dimension.

Figure 10 shows the shut-down transient in the sub-island 1 for the two generators in frequency regulation. One can see a small power oscillation (lower than 20 kW), taking place symmetrically between the two machines.

5 Conclusions and future work

At the beginning of the activity to which this paper belongs, the research question was whether or not is it feasible to use the Modelica language and Modelica simulation tools to handle nation- and continental-wide electromechanical power system models. Over the last year, we reached an affirmative conclusion, though there is clearly work to be done to speed up the code generation phase, which is still too long for many application contexts.

More in detail, we could prove the feasibility of using 100% Modelica models for the simulation of transients in systems of national and continental size, albeit currently with very simple generator and controller models. The only exception is topological analysis, which will arguably be better handled by external C code, possibly re-using legacy code that performs the same task.

The simulation times we observed, particularly when using the variable step-size sparse DAE solver IDA, are acceptable, and are certainly amenable to further improvements as the implementation of that solver in OpenModelica is streamlined and optimized. On the other hand, there is still much work to do in order to reduce the time for code generation by at least one order of magnitude. Development activities are under way on the OpenModelica compiler to achieve this goal, most notably a new, much faster front-end, as well as code generation algorithms that are optimized for the sparse DAE solver. We also evidenced the need for further improvements as for the model initialisation and the event handling.

It is worth noticing that we could carry out all the re-

DOI 10.3384/ecp17132227

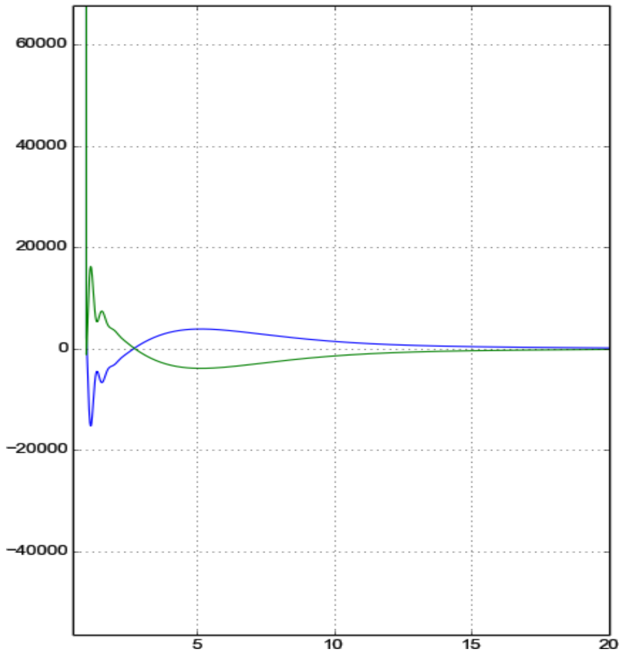

Figure 10. Active power of generators in sub-island 1.

search activity entirely within the OpenModelica framework, particularly after several improvements were made to the compiler front-end, back-end, and simulation runtime.

Given the positive outcome of this first one and a half year of ground-breaking work, the authors believe that some more specific investment in the development of the OpenModelica tool for this type of applications could lead to much better performance than what is reported in this paper. Even if the performance of domain-specific tools may not be fully reached, the added value brought in terms of flexibility and openness by the use of the Modelica object-oriented modelling framework, as well as by the use of open-source tools like OpenModelica, makes this research activity worth to be further pursued.

6 Acknowledgements

The authors gratefully acknowledge the financial support of CESI S.p.A. and RTE for supporting this research work. They also want to thank the entire development team of OpenModelica for their support and hard work, which made these developments possible.

References

F. Casella. Simulation of large-scale models in Modelica: state of the art and future perspectives. In *Proc. 11th International Modelica Conference*, pages 459–468, Versailles, France, 2015.

F. Casella, A. Bartolini, S. Pasquini, and L. Bonuglia. Object-oriented modelling and simulation of large-scale electrical power systems using Modelica: a first feasibility study. In *Proc. 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 6298–6304, Florence, Italy, 2016.

ENTSO-E. ENTSO-E policy paper: Future TSO coordination for Europe. Technical report, ENTSO-E, 2014.

ENTSO-E. ENTSO-E work programme 2015 through 2016. Technical report, ENTSO-E, 2015.

R. Franke and H. Wiesmann. Flexible modeling of electrical power systems – the Modelica PowerSystem library. In *Proc. of the 10th International Modelica Conference*, pages 515–522, Lund, Sweden, 2014.

C. Kral and A. Haumer. Modelica libraries for DC machines, three phase and polyphase machines. In *Proc. 4th International Modelica Conference*, pages 549–558, Hamburg, Germany, 2005.

L. Vanfretti, M. LI, T. Bogodorova, and P. Panciatici. Unambiguous power system modeling and simulation using Modelica tools. In *2013 IEEE Power & Energy Society General Meeting*, 2013.

L. Vanfretti, T. Bogodorova, and M. Baudette. A Modelica power system component library for model validation and parameter identification. In *Proc. 10th International Modelica Conference*, pages 1195–1203, Lund, Sweden, 2014.

M. Zhang, M. Baudette, J. Lavenius, S. Løvlund, and L. Vanfretti. Modelica implementation and software-to-software validation of power system component models commonly used by nordic TSOs for dynamic simulations. In *Proc. 56th SIMS Conference on Simulation and Modelling*, pages 105–112, Lund, Sweden, 2015.