Mixed phasor and time domain modelling of AC networks with changeover management

Hakan Dogan Parildar[*], Alberto Leva[†]
Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria
Via Ponzio 34/5, 20133, Milano, Italy

Abstract

Simulation studies on AC electric networks may comprehend periods of quasi-stationary operation and rapid transients. The adoption of a phasor-based approach results in high simulation efficiency, but is limited to the first of the two situations above, while for the second, time domain models are required. For system studies where both situations have to be simulated, a modelling paradigm is thus required that can join the two approaches in all the described components, and by which the simulator of an entire network can be endowed with the capability of moving back and forth from a phasor to a time domain system description automatically, taking care of the proper re-initialisations when necessary. In this paper we propose a possible solution, and apply our ideas by presenting the first *nucleus* of a Modelica library designed along them. We also show some simulation examples to support the validity and practical convenience of the proposal.

Keywords: AC networks, phasor models, time domain models, efficient simulation.

1 Introduction

Dynamic simulation is nowadays of great importance for both the design and the operation of electric networks [5]. Indeed, the availability of reliable simulation models is often enabling for the realisation of several functionalities of "smart" grids, like those described in works such as [9].

When addressing system studies – for example, but in principle not exclusively, for control synthesis purposes – it is frequently necessary to simulate long periods of (quasi-)stationary operation, interspersed with abrupt events. These two situations allow (and in some sense call) for different modelling paradigms, however. Stationary operation is well described by phasor models, to the advantage of simulation efficiency. Events like for example the opening of a switch, on the other hand, require time-domain modelling, which is far more intensive from the computational standpoint. An intermediate case is given by "quasi-stationary" regimes, where the phasor approach can be extended by means of the so-called "swing equation", which (simplifying for brevity) turns the algebraic phasor framework to a dynamic one, introducing machine angles as the state variables.

Focusing on the two extreme cases above, it would be highly beneficial and desired that a network simulation model could switch back and forth between a phasor and a time domain description, possibly in an automatic manner, and requiring as small an effort as possible on the part of the analyst. Quite expectedly, a significant research effort is being spent on the matter, but nonetheless some questions are still open, especially if a strict application of the object-oriented paradigm is required—i.e., more specifically, if all the encountered modelling issues are to be managed at the level of the individual components.

In this paper, we formulate a proposal to address the problem just mentioned, and prove its viability by presenting the first *nucleus* of a Modelica library, combining time domain and phasor modelling, developed along the devised approach.

The paper is organised as follows. Section 2 briefly discusses some related work, also to motivate the presented research, while Section 3 introduces the proposed approach, focusing on the automatic changeover problem. In Section 4 the library *nucleus* is outlined, and Section 5 reports a simulation example to demonstrate the viability and usefulness of the approach. Section 7 points out and synthetically discusses some open issues, while Section 8 draws some conclusions and sketches out future research.

[*]Former graduated student at the Dipartimento di Elettronica, Informazione e Bioingegneria

[†]Corresponding author, alberto.leva@polimi.it

2 Literature review and motivation

The need for modelling in the context of (AC) electric networks and their management has been recognised since long time ago [6]. The introduction of renewable energies and distributed generation has then increased the interest on the matter in the last decades [10], and further impulse to the mentioned research has been coming from smart grids [8].

Nowadays, simulation models of electric networks are also often combined with those of the generators' prime movers [1] to form multi-physics, multi-scale and potentially large overall models, for which the object-oriented paradigm of Modelica is particularly suited. In such cases, however, the electric part of the model is often the bottleneck for simulation efficiency, and the reason for that is structural.

In extreme synthesis, in fact, an AC network can be modelled at three levels. The most efficient one from the computational standpoint is provided by phasors [2]: this framework allows to write an algebraic model, that however is valid only in the hypothesis of a single, constant frequency for all the network. Small fluctuations of "local" frequencies are allowed by the so-called "swing equation" formalism,see e.g. [7], the application of which leads to dynamic component models, having machine angles as state variables.

Quite intuitively, the idea of using phasor models in Modelica is not new, but to the best of the authors' knowledge, to date no attempt was made to have phasor and time domain descriptions *co-exist* at the *component* level. For example, in [3] the idea of coupling phasor-domain and "transient" models is introduced, but the connection between the two relies on causal signals, making it difficult to represent it at the individual component level, especially for what concerns the domain changeover. Another interesting paper is [4], where however no changeover to time domain is considered, and the possibilities of phasor-based modelling are exploited via a convenient use of the swing equation.

As such, even a minimal literature analysis like that reported indicates that the problem addressed herein is of both theoretical and practical interest, and that the attempted solution has some novelty characteristics.

3 Mixed time-domain and phasor modelling

As anticipated, phasor models in Modelica are not a novelty [3, 4], and as such, the proposals that we are making with this paper are to be integrated in the *scenario* depicted by works like the ones just quoted.

However, this research has some specific peculiarities, a discussion on which (and the consequently proposed solution) provide the main contributions of this paper. Specifically, three aspects are herein addressed:

- creating models that contain *both* a phasor and a time domain description of a given component,
- managing the transition (changeover) between the two at the component level,
- managing the *decision* to make a changeover at the overall model level.

As can be easily guessed, the main problem in structuring the required models is to preserve object orientation, the benefits of which are apparently not to be justified here, despite some facts to be handled are not confined to a single component—most notable, in this respect, is the changeover decision.

The rest of this section, organised along the items above, presents the proposed solution and provides motivations for it, also in a view to stimulating further discussions and improvements.

3.1 Component-level modelling

We now consider the problem of mixing phasor and time domain descriptions in the same component model, and of managing at that level the changeover between the two.

First, suitable physical (i.e., in this case, electrical) connectors are necessary. To this end, a positive phasor and time domain pin is straightforwardly defined (we omit trivial code like the inclusion of the `Modelica.SIunits` package) as

```
connector ptPin "phasor/time positive
    pin"
      Voltage Vre "V phasor, R part";
      Voltage Vim "V phasor, I part";
      Voltage v   "v(t), time domain";
 flow Current Ire "I phasor, R part";
 flow Current Iim "I phasor, I part";
 flow Current i   "i(t), time domain";
end ptPin;
```

Listing 1: connectors.

and a negative one is created in the same way. Then, each component has to contain the constitutive equations for both the phasor and the time domain description, and to this end, we have to distinguish between passive components and active ones—i.e., generators.

DOI 10.3384/ECP14096533

Taking the `Inductor` component as a representative example for the former type, and denoting by `a` and `b` its positive and negative pin, respectively, the time domain description is

```
0       = a.i + b.i;
a.v-b.v = L*der(a.i);
```

Listing 2: inductor, time domain equations.

while the phasor one reads

```
0             = a.Ire+b.Ire;
0             = a.Iim+b.Iim;
a.Vre-b.Vre   = (L*freqHz*2*.pi)
                *sqrt(a.Ire^2+a.Iim^2)
                *cos(atan2(a.Iim, a.Ire)
                     +pi/2);
a.Vim-b.Vim   = (L*freqHz*2*pi)
                *sqrt(a.Ire^2+a.Iim^2)
                *sin(atan2(a.Iim, a.Ire)
                     +pi/2);
```

Listing 3: inductor, phasor equations.

Finally, the code for initialising the time domain equation when needed is

```
if flag_for_reinit then
   when {TimeDom} then
      reinit(a.i,
        sqrt(pre(a.Iim)^2+pre(a.Ire)^2)
        *sin(2*pi*freqHz*time
        +atan2(pre(a.Iim),pre(a.Ire))))
   end when;
end if;
```

Listing 4: inductor, time domain (re-)initialisation.

where `TimeDom` is a flag indicating that the entire model is simulated in the time domain.

Coming to generators, the relevant equations for an ideal sine voltage one are shown below.

```
0             = a.Ire+b.Ire;
0             = a.Iim+b.Iim;
cos(phase)*V  = a.Vre-b.Vre "Re of V";
sin(phase)*V  = a.Vim-b.Vim "Im of V";
0             = a.i + b.i;
if TimeDom then
   a.v-b.v    = V*sin(2*pi*freqHz*time
                      +phase);
else
   a.v-b.v    = 0;
end if;
```

Listing 5: sine voltage generator equations.

At the component level, changeover is thus managed by acting in a coordinated manner on passive and active components, in a view to avoiding unnecessary events and enhance simulation speed. It can be in fact noticed that when the phasor model is in use, the time domain voltage is just zeroed. This makes the state variable `a.i` in Listing 2 simply decay to zero, avoiding conditional equations and limiting the triggered events to the bare minimum (the transition of `TimeDom` from true to false, or *vice versa*). Of course, said decay to zero is meaningless as simulation result, which is however indicated by thevalue of `TimeDom`.

The only *caveat* with the proposed solution is that the currents of inductors in series – or, dually, the voltages of capacitors in parallel – may be treated by the tool at hand as a single entity. The parameter `flag_for_reinit` in Listing 4 has the purpose of avoiding inconsistencies in such cases. At present, it is the user's responsibility to ensure that in each set of inductors in series, or capacitors in parallel, one and only one has that parameter set to true. Apart from this, however, object orientation (or more specifically, the independence of a component description of how it is connected to the others) is fully preserved.

3.2 System-level modelling

We now move to the problem of managing the changeover between the time domain and the phasor representation at the level of the entire simulation model, i.e., of the network—in synthesis, thus, we specify how the `TimeDom` flag is handled.

Switching from phasor to time domain is generally the consequence of some abrupt event known to at least one component, like for example a closing or opening switch. It is thus assumed that in such a case, the affected component directly sets the flag to true, causing the changeover instantaneously.

A bit more difficult is conversely the reverse changeover, since to make it feasible, *all* the currents and voltages need settling to a sinusoidal regime. The decision is in this case taken on the basis of local signalling from the components, and of a unanimity verification mechanism at the system level.

3.2.1 Local signalling of sinusoidal regime

Also in this case, like it was for the component-level changeover management shown in Section 3.1, the main goal of model design is to avoid unnecessary events, for efficiency reasons.

Suppose therefore that we need to detect if a certain variable $x(t)$ – voltage or current – is "sufficiently" close to a sinusoid with frequency `freqHz`, assumed for the moment of zero mean for simplicity (releasing this is straightforward, see later on). Filtering $x(t)$ through the continuous-time transfer function

$$F(s) := \frac{X_F(s)}{X(s)} = \left(\frac{1}{\pi f_o} \frac{s}{1 + \frac{s}{\pi f_o} + \frac{s^2}{(2\pi f_o)^2}} \right)^{n_F} \quad (1)$$

produces an output $x_F(t)$ equal to the input $x(t)$ if and only if the frequency of the latter is exactly f_o, which is apparently set to `freqHz`.

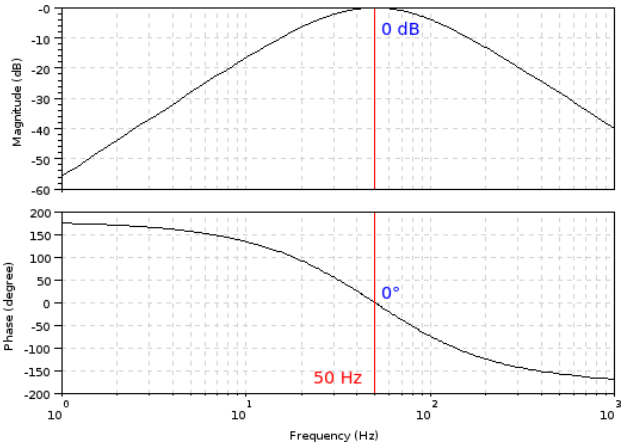

Figure 1: frequency detector – Bode plots of $F(j\omega)$.

Parameter n_F is used to enhance the attenuation of the frequency response $F(j\omega)$ as the input frequency moves away from f_o, and a value of two (or three at most) proved enough in practice. The operation of $F(s)$ is shown by the Bode plots of Figure 1, obtained with $f_o = 50Hz$ and $n_F = 2$.

The output of $F(s)$ in (1) is then used, together with its input, to form the signal

$$s(t) = \frac{1 + x_f(t)^2}{1 + x(t)^2} \quad (2)$$

which is structured so as to inherently avoid division by zero errors, and finally $s(t)$ is lowpass-filtered by the unity-gain first-order block

$$D(s) := \frac{Y(s)}{S(s)} = \frac{1}{1 + s\frac{k_f}{2\pi f_o}} \quad (3)$$

where parameter k_f is used to control the achieved smoothing (a value of ten is a good default). As a result, $y(t)$ will signal the required condition on $x(t)$ by taking a value very close to the unity, with small fluctuations.

Comparing the value and the time derivative of $y^2(t)$ to suitable thresholds, where squaring the signal is to avoid the events that would be generated if its absolute value were conversely taken, is therefore a means to detect that $x(t)$ is close enough to a sine wave with frequency `freqHz`.

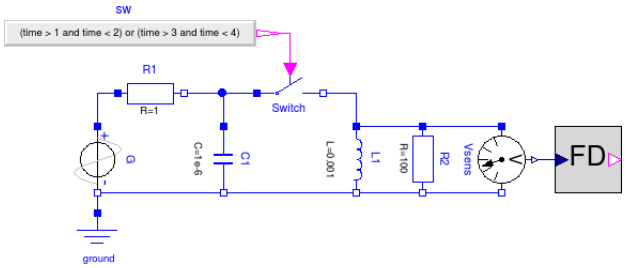

Figure 2: frequency detector test – Modelica diagram.

To show the efficacy of the proposed technique, and also its autonomy with respect to the rest of the proposed modelling paradigm, Equations (1) through (3), together with the mentioned thresholding mechanism, were turned into a `FreqDetector` block, that is used in the model of Figure 2 together with components from the `Modelica.Electrical` library (its use in the presented one, with the actual introduction of phasor modelling, will be illustrated later on).

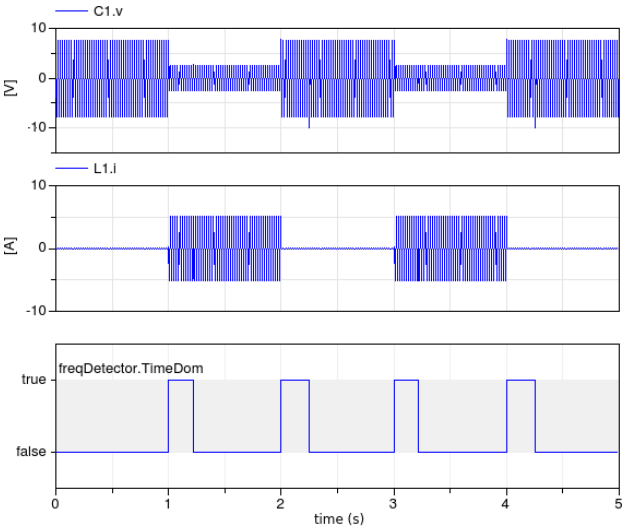

Figure 3: frequency detector test – simulation results.

Figure 3 shows a sample simulation test. Detailed figures are inessential for its purpose, apparently, but as can be seen, the need for time domain modelling as caused by the switch closing and opening is detected correctly, especially for the transition toward (the possible use of) phasor mode.

Recall that the symmetric transition is in any case guaranteed by the locally originated signalling, see

DOI 10.3384/ECP14096533

Section 3.1 above. This is because the phasor to time domain transition must be *instantaneous* to preserve accuracy, while if the time to phasor domain on is delayed with respect to the time when it is acceptable, the only relevant effect is some waste of CPU time.

It is worth noticing that the 5s simulation of Figure 3 involves only 14 state events, which would apparently not be true if the possibility of switching to phasor mode were identified based on zero crossing counters, or similar methods. Note also that the proposed technique introduces some additional state variables, but these pertain to *linear, time invariant, causal* blocks *cascaded* to the physical model. The resulting simulation overhead is thus modest, and in any case much lower than that of zero-crossing or similar methods.

The default values chosen allow for a safe transition toward phasor modelling, without bounces and within a reasonable time. In the presence of an essentially constant-frequency behaviour interspersed with abrupt events that make the frequency content of the involved signals radically different from the (quasi-)stationary one, this is a good compromise between fast transition to phasor mode (which undoubtedly favours simulation efficiency) and possibly undue switching of the two modes (which conversely may be detrimental owing to re-initialisations). Also, the presented detection method is inherently normalised, since so are all the involved quantities (except times, of course). This makes the selected default values for the involved parameters valid in a wide operating range, and for an equally wide variety of network physical parameters.

To manage a possible nonzero average of $u(t)$, finally, the proposed filtering path is implemented as a series of transfer function blocks from the Modelica Standard Library, and signal $s(t)$ in (2) is formed by taking the output of the first block in the place of $u(t)$. This ensures that the average of the signal taken as input settles to zero with a dynamics comparable to that of the transients superimposed to its steady-state sinusoidal behaviour, and at the same time preserves the exploitation of the unity-magnitude and zero-phase frequency response values at the sought frequency so as to realise the envisaged detection system.

3.2.2 System-level handling of `TimeDom`

To manage the `TimeDom` flag at simulation time, denote respectively with N_{pt} and N_{tp} the number of elements entitled to cause a changeover toward time domain mode (typically, switches), and that of elements the voltage across which is to be checked to approve the reverse transition.

From a conceptual standpoint this set could well be the totality of the present passive components, but for optimisation reasons the user can be allowed to introduce "frequency probes", based on the described `FreqDetector` component, only where deemed necessary.

At the present state of the library development, this architectural choice is still open: most likely, however, based on the experience that is being gathered, the final solution will be to distinguish a "basic" mode, where any passive component has a detector, and an "expert" one, where the user is free to configure the mechanism at his/her best convenience.

In any case, assuming – in principle, as the implementation described in the following section is different for efficiency reasons – two boolean vectors `P2T` and `T2P`, of length N_{pt} and N_{tp} respectively, to be declared at the outermost model level, and omitting trivial details on the inner/outer manner they are managed, the following procedure for handling `TimeDom` is adopted.

1. The simulation starts out in time domain mode, for the safe side (possibly, in the future, unless differently specified in the "expert" mode). All the elements of `P2T` and `T2P` are conveniently initialised (at present, to false).

2. All entitled elements manage their local time domain flag based on the contained `FreqDetector` element, and the system-level `T2P` vector collects them all.

3. If at the system level time domain mode is in use, and all the elements of `T2P` are true, then the system switches to phasor domain mode.

4. If at the system level phasor mode is in use, any transition to true of at least one element of `P2T` causes a changeover to time domain mode, with the required re-initialisations.

3.2.3 Modelica implementation

The solution just described serves the intended purpose, and the realised one is totally equivalent from a conceptual and functional standpoint.

However, if said solution were implemented literally, some deviation from a totally object-oriented setting would be involved, since any component participating in either of the two mentioned boolean vectors, would have to contain suitable parameters to indicate

which position in said vectors pertain to it. The responsibility of setting those indices correctly would stand with the user, being possibly complex and cumbersome to manage for large models.

In addition, and most important, even if the management of the mentioned indices were somehow automated, some model connections would in this way be realised, that do not fall under a proper connector abstraction.

To overcome this relevant problem, the described solution is therefore implemented as follows. First, a `ChangeoverMgmt` connector is defined as shown below.

```
connector ChangeoverMgmt
 flow Integer Ntp "# of T->P voters";
 flow Real ForceP2T;
 flow Real AllowT2P;
 Modelica.SIunits.Frequency freqHz;
 Boolean TimeDom;
 Real dum "Squelch balancing warnings";
end ChangeoverMgmt;
```

Listing 6: the `ChangeoverMgmt` connector.

Each component participating in the changeover decision (i.e., each model of reactive elements or of commuting ones like switches), and also each generator, is endowed with such a connector, named in the following `C`; also, all those models take the `TimeDom` flag and the (nominal) frequency from that connector.

Reactive components (the *Inductor* model is an example) furthermore contain the code

```
 C.Ntp       = -1;
 C.ForceP2T  =  0;
 ...
 C.AllowT2P = if C.TimeDom and not
     FD.TimeDom
                then -1 else 0;
```

Listing 7: inductor, changeover management.

The first line reported in Listing 7 provides the `Supervisor` component, described later on and that also has a `ChangeoverMgmt` connector to which those of all network elements are connected, with the number of those elements that vote for the time domain to phasor mode transition.

The second line means that the component, given its role, is not entitled to force a transition from phasor to time domain model.

The last reported line, finally, casts the vote when this is required.

Models of commuting components (like switches) conversely contain the code

```
 parameter Real Tsw   = 0.01;
 parameter Real thrsw = 0.01;
 ...
 Real xsw(start=1);
 ...
 xsw+Tsw*der(xsw)= if control
                   then 1 else 0;
 C.ForceP2T      = if control
                     and xsw<1-thrsw
                     or not control
                     and xsw>thrsw
                   then -1 else 0;
```

Listing 8: inductor, changeover management.

When the `control` input (the switch command) commutes, the introduced dynamic variable `xsw` is used to generate a square pulse with a minimum of state events, and this is in turn used – see the last line in Listing 8 – to signal that the component intends to force a transition from phasor to time domain mode.

Finally, the *Supervisor* component is implemented as per Listing 9 below.

```
model Supervisor
 parameter Frequency fo=50;
 Interfaces.ChangeoverMgmt C;
 Boolean P2T,T2P;
equation
 C.dum    = 0;
 P2T      = C.ForceP2T>0.5;
 T2P      = C.AllowT2P>C.Ntp-0.5;
 C.freqHz = fo;
algorithm
  when T2P and not P2T then
       C.TimeDom := false;
  end when;
  when P2T then
       C.TimeDom := true;
  end when;
initial equation
  C.TimeDom = true;
end Supervisor;
```

Listing 9: supervisor.

The initial equation makes the simulation start in time domain, as specified in Section 3.2.2. Then, thanks to the `flow` connections, variables `ForceP2T` and `AllowT2P` respectively sum the forcing to time domain mode requests, and the permission for phasor mode votes. based on that, when `ForceP2T` is at least 0.5, then at least one component is forcing time domain mode.

DOI 10.3384/ECP14096533

Analogously, when `AllowT2P` exceeds the number of voters (collected in the `Ntp` connector variable) minus 0.5, then all said voters are permitting the transition toward phasor mode. Finally, the two `when` clauses in the `algorithm` section manage the `TimeDom` flag, triggering events only when necessary.

4 The library structure

As anticipated, the presented ideas were applied to create the first *nucleus* of a Modelica library for mixed phasor and time domain modelling of electric networks.

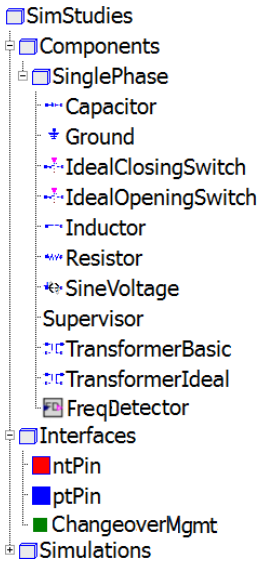

Figure 4: the library (*nucleus*) structure.

Said *nucleus* is structured as outlined in Figure 4, and to date comprises only single-phase components, connectors, and the `Supervisor` block to be included in the realised models so as to implement the necessary network-wide declarations, and the management of the involved variables.

Plans are of course to extend the *nucleus* to form a complete library, including more articulated components such as non ideal generators, the representation of multiphase elements, and the like. Also, care will be taken to integrate the consequent future work with (at least) the similar ones shown in the references quoted above in Section 2.

5 An illustrative simulation example

We now shows a simple simulation example to demonstrate the operation of the proposed modelling framework, and specifically of the changeover management.

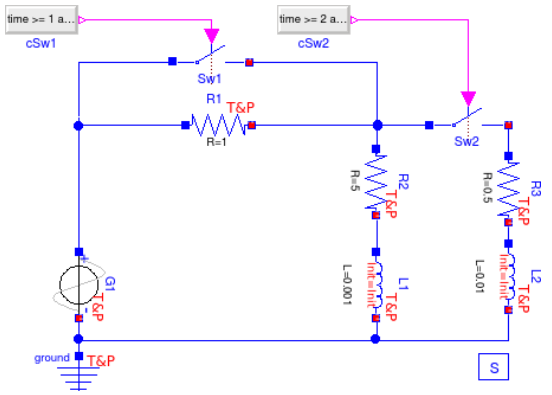

Figure 5: illustrative simulation example – the used network model.

The example refers to the small network model depicted in Figure 5. The switch `Sw1`, initially closed, is opened at $t = 1\,s$ and re-closed at $t = 3\,s$. The switch `Sw2`, also initially closed, is cycled at $t = 2\,s$ and $t = 4\,s$. The simulation run duration is $5\,s$.

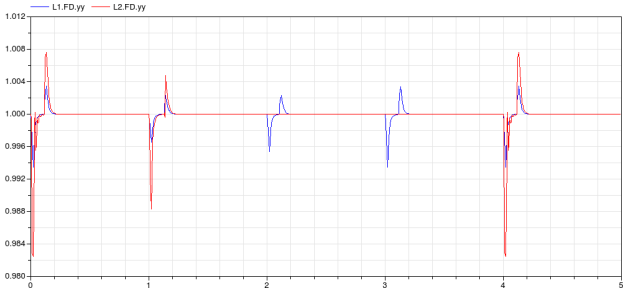

Figure 6: illustrative simulation example – y signals in the frequency detectors of the two inductors.

Figure 6 shows the $y(t)$ variables – see (3) – in the frequency detectors of the two inductors. Recall that those variables are meant to indicate, by assuming a nearly constant unity value, the settling of the locally measured frequency to `freqHz` (here set to 50 Hz). As can be seen, the expected effect is obtained, and the normalised nature of the involved signals produces comparable transients also in the presence of different values for the components' electric parameters.

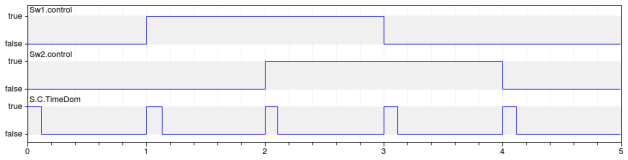

Figure 7: illustrative simulation example – boolean flags.

In Figure 7, the relevant boolean flags at the system level are instead represented. The changeover mecha-

nism catches the switch events, passing to time domain instantaneously, while the time domain periods are not equal, correctly depending on the transient behaviour of the monitored electric variables.

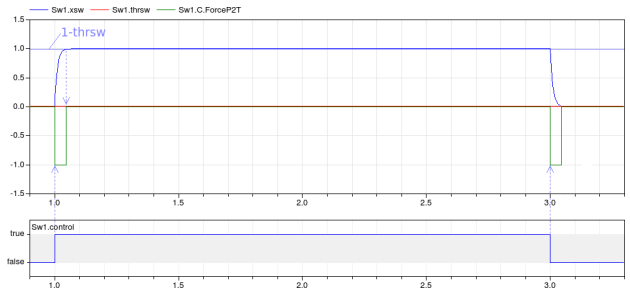

Figure 8: illustrative simulation example – forcing time domain on the part of a switch.

Figure 8 illustrates how a commuting component forces the transition to time domain mode by means of `xsd`, and the role of the threshold and the time constant of its dynamics in determining the width of the generated pulse. Note that the rising edge of that pulse is structurally synchronous to the switch command, which is consistent with the defined specifications.

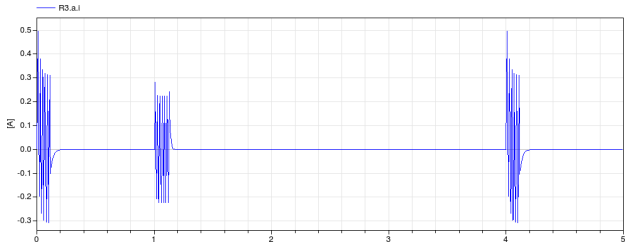

Figure 9: illustrative simulation example – current in `R3`, time domain representation.

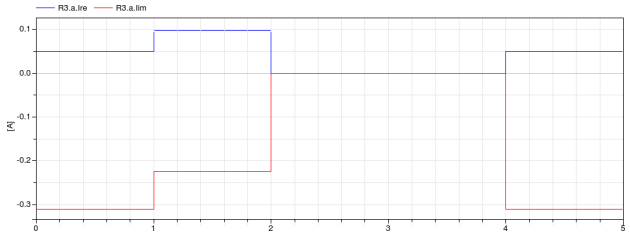

Figure 10: illustrative simulation example – current in `R3`, phasor representation.

Figures 9 and 10 show the behaviour of the current flowing through resistor `R3`, viewed respectively in the time domain and as a phasor (for which the real and the imaginary part are plotted). Notice that when the simulation switches back to phasor mode after a period in time domain mode triggered by a switch event, the time domain signal has actually settled back to a sinusoidal behaviour, making the changeover correct.

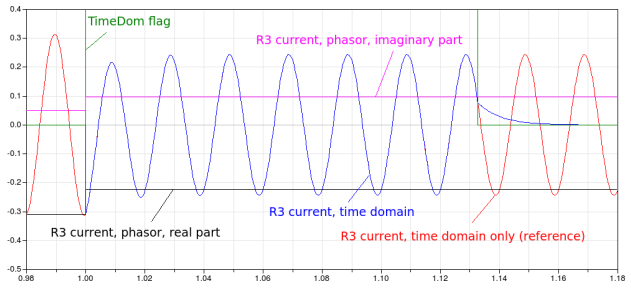

Figure 11: illustrative simulation example – current in `R3`, changeover detail.

Finally, Figure 11 demonstrates the changeover and re-initialisation mechanism, comparing the phasor and time domain representations for the current through `R3` with a reference obtained by forcing the simulation to take place entirely in the time domain. Observe the (re-)initialisation of the time domain representation of the shown variable, triggered instantaneously by the changeover to time domain mode. Note also how the time domain representation of the same variable settles to zero after the changeover to phasor mode occurs, allowing the solver to exploit variable-step capabilities to the advantage of efficiency. As can be verified, then, taking the time or phasor domain signals as the valid ones depending on `TimeDom`, conveys all the required information.

6 A simulation example on efficiency

Although on this matter work is still in progress, we report an example showing the simulation efficiency gained with the proposed approach.

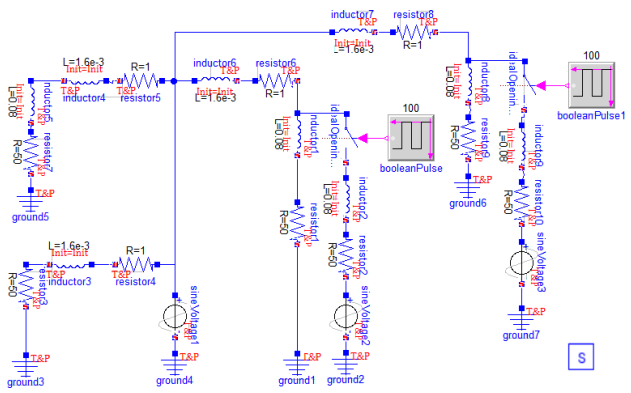

Figure 12: simulation example on efficiency – the used network model.

The example refers to the network model depicted

DOI 10.3384/ECP14096533

in Figure 12, apparently more complex than that of Figure 5 above. The network was assembled with components from the presented library, and a reference model for it was created with the equivalents from the Modelica Standard Library, limiting of course the scope to time domain modelling. Various simulations were then performed with a duration of 10^4 seconds, cycling the switches at regular intervals for a variable number of times.

Cycles	CPU time	Steps	F-evals	J-evals
800	161	26276873	52580944	16847
600	155	26038476	52097092	12594
400	164	26036633	52086819	8404
200	161	26103648	52214194	4259
100	160	26177418	52358292	2131
80	168	26095949	52194637	1699
40	162	26126749	52254882	853

Table 1: simulation example on efficiency – results with the Modelica Standard Library.

Cycles	CPU time	Steps	F-evals	J-evals	Events
800	134.00	1717702	5968550	759422	59257
600	97.60	1246309	4293129	538653	42139
400	67.80	857706	2980426	379132	29580
200	33.70	429941	1493272	189871	14758
100	16.90	215300	747528	95007	7424
80	13.90	173998	605417	77192	6008
40	7.12	90491	316665	40846	3207

Table 2: simulation example on efficiency – results with the proposed models.

Table 1 shows the results obtained with the Modelica Standard Library. The first column reports the number of switch cycles in the simulation, while the others contain the typical computational load statistics. Table 2 conversely shows the results achieved with the proposed models, and has a further column that reports the number of generated state events (that apparently is zero in the previous case).

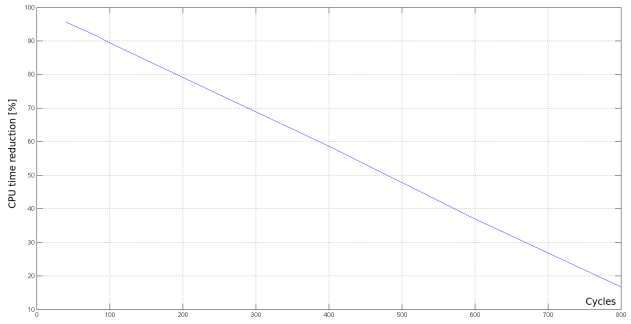

Figure 13: simulation example on efficiency – CPU time percent reduction versus switch cycles.

As can be seen, the efficiency improvement is quite evident in terms of CPU time, for which the gain is represented graphically in Figure 13, number of steps and F-evaluations, although (owing most likely to the numerous re-initialisation) there is quite an increase of the Jacobian evaluations.

7 Open issues

As stated right from the introduction, the work presented in this paper is still in progress. We have reached sufficient certainties on the viability of the approach, with particular reference to the possibility of managing the representation changeover within a solid object-oriented setting, and without triggering undue event hauls. However, more than one relevant issue stands still open, and this section is devoted to a brief overview on them.

A first point concerns the quantification of the obtained advantages. Here some figures referring to efficiency were given, but no doubt, more extensive efficiency assessment campaigns are in order, and to this end, testing is underway with larger models. At a first glance, the impression we are getting is that the number of simulation steps reduces more or less proportionally to the fraction of the overall simulation run that can be done with phasors, which is in reasonable accordance with intuition. On the other hand, the number of Jacobian evaluation exhibits more peculiar behaviours, on which deeper investigations are also to be performed. Also, it is advisable that future studies address possible interactions with the solution algorithm, of the variable-step type for apparent reasons: at present, in fact, to date all the tests were done with DASSL only.

A second point concerns the parameters involved in the changeover management machinery. Despite the use of normalised signals has proven effective, in fact, we still observe some residual effects of the threshold values, on which further studies are consequently in order. In particular, it would be useful to somehow relate said thresholds, and also the value of the time constant for the dynamics of `xsd`, to the desired tradeoff between accuracy and efficiency improvement, quantified (for example) on the basis of the additional time spent simulating in time domain mode beyond necessary.

Moving from efficiency to model manipulation and structuring issues, a first point concerns the re-initialisation flags. At present these are to be managed by the user, but apparently could be set automati-

cally if the tool manipulation chain were somehow instructed to recognise and treat series/parallel connections. In principle, leaving the flag responsibility to the user does not seem a big problem, unless when dealing with large models where connections are changed frequently—not too realistic a case. Nonetheless, also on this point further research is advisable.

A more relevant issue is relative to the use and abstraction of non-physical connectors like the `ChangeoverMgmt` one. Here, the quest for efficiency in fact led to somehow abuse flow variables (by the way, the authors would like to thank the colleagues Francesco Casella and Victorino Sanz for ideas and useful suggestions in this respect). However, with the mechanism here adopted, more articulated voting mechanisms than the realised ones, would be cumbersome and possibly impossible to implement. In contexts like that addressed herein such mechanisms may not be necessary, but the experience reported in this work suggests that some way to tailor the connector semantics would be highly desirable—a small suggestion for discussions on future versions of the language.

Finally, although this is undoubtedly simpler and less important from a conceptual point of view, some post-processing tool would be needed that uses the produced results (phasors and time domain variables, together with `TimeDom` to say which one is significant at which time instant) to reproduce and present all the simulated quantities entirely in the time domain, for the user convenience.

8 Conclusions and future work

The problem of creating models of AC networks joining the phasor and the time domain approach, and switching from one another automatically during a simulation run, was addressed. A solution was proposed, having some peculiar features in particular as for the changeover mechanism, and motivations for the adopted choices were provided. The approach was put to work by creating the first *nucleus* of a Modelica library, and a simulation example – referring to a very simple case given the purpose of this paper – was shown to back up the viability and usefulness of the proposal.

Future work will be essentially aimed at tackling the issues pointed out in Section 7. In addition, the integration with models for the process sections of generators (typically, their prime movers) will be addressed, in a view to better exploit the efficiency improvements yielded by the proposed approach. Finally, given the interest on the considered modelling and simulation domain, the authors hope that this proposal can stimulate discussions, further ideas, and collaborations.

References

[1] F. Casella and A. Leva. Modelling of thermo-hydraulic power generation processes using Modelica. *Mathematical and Computer Modelling of Dynamical Systems*, 12(1):19–33, 2006.

[2] C.A. Desoer and E.S. Kuh. *Basic circuit theory*. McGraw-Hill, New York, NY, 1966.

[3] O. Enge, C. Clauß, P. Schneider, P. Schwarz, M. Vetter, and S. Schwunk. Quasi-stationary AC analysis using phasor description with Modelica. In *Proc. 5th International Modelica Conference*, pages 579–588, Vienna, Austria, 2006.

[4] A. Haumer, C. Kral, J.V. Gragger, and H. Kapeller. Quasi-stationary modeling and simulation of electrical circuits using complex phasors. In *Proc. 6th International Modelica Conference*, pages 229–236, Bielefeld, Germany, 2008.

[5] W.W. Hogan. Markets in real electric networks require reactive prices. In M. Einhorn and R. Siddiqi, editors, *Electricity transmission pricing and technology*, pages 143–182. Springer, Dordrecht, The Netherlands, 1996.

[6] M. Huneault and F.D. Galiana. A survey of the optimal power flow literature. *IEEE Transactions on Power Systems*, 6(2):762–770, 1991.

[7] D.P. Kothari. *Modern power system analysis*. Tata McGraw-Hill Education, Noida, India, 2003.

[8] V. Liberatore and A. Al-Hammouri. Smart grid communication and co-simulation. In *Proc. Energytech*, pages 1–5, 2011.

[9] J.A.P. Lopes, C.L. Moreira, and A.G. Madureira. Defining control strategies for microgrids islanded operation. *IEEE Transactions on Power Systems*, 21(2):916–924, 2006.

[10] J.R. Ubeda and M.A.R. Rodriguez Garcia. Reliability and production assessment of wind energy production connected to the electric network supply. In *IEE Proceedings – Generation, Transmission and Distribution*, volume 146, pages 169–175, 1999.

 DOI 10.3384/ECP14096533