SoftwareX 10 (2019) 100253

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

DPsim—A dynamic phasor real-time simulator for power systems

Markus Mirz *, Steffen Vogel, Georg Reinke, Antonello Monti

Institute for Automation of Complex Power Systems, RWTH Aachen University, Aachen, Germany

ARTICLE INFO

Article history:
Received 17 December 2018
Received in revised form 12 April 2019
Accepted 21 May 2019

Keywords:
Real-time simulation
Dynamic phasors
Co-simulation

ABSTRACT

DPsim is a real-time capable solver for power systems that operates in the dynamic phasor and electromagnetic transient (EMT) domain. This solver primarily targets co-simulation applications and large-scale scenarios since dynamic phasors do not require sampling rates as high as EMT simulations do. Due to the frequency shift introduced by the dynamic phasor approach, the sampling rate and rate of data exchange between simulators can be reduced. DPsim supports the Common Information Model (CIM) format for the description of electrical network topology, component parameters and power flow data and it is closely integrated with the VILLASframework to support a wide range of interfaces for co-simulation. Simulation examples demonstrate the accuracy of DPsim and its real-time performance for increasing system sizes.

Code metadata

Current code version	v1.0.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2018_244
Legal Code License	GPLv3
Code versioning system used	git
Software code languages, tools, and services used	C++, python
Compilation requirements, operating environments & dependencies	DPsim can be compiled for Linux, OSX and Windows. The main dependencies are: gcc, Eigen, Python, CIM++, VILLASnode, Sundials. A Dockerfile with all dependencies is included in the repository.
If available Link to developer documentation/manual	https://fein-aachen.org/projects/dpsim/
Support email for questions	mmirz@eonerc.rwth-aachen.de

Software metadata

Current software version	v1.0.0
Permanent link to executables of this version	https://hub.docker.com/r/rwthacs/dpsim
Legal Software License	GPLv3
Computing platforms/Operating Systems	Linux docker container
Installation requirements & dependencies	Only a docker installation is required.
If available, link to user manual	https://fein-aachen.org/projects/dpsim/
Support email for questions	mmirz@eonerc.rwth-aachen.de

1. Motivation and significance

DPsim introduces the dynamic phasor (DP) approach to real-time power system simulation. The motivation is to remove the requirement of proportionality between the simulation time step

* Corresponding author.
E-mail address: mmirz@eonerc.rwth-aachen.de (M. Mirz).

https://doi.org/10.1016/j.softx.2019.100253

and the highest frequency of simulated signals. Especially for power electronics and geographically distributed real-time simulation, this is an interesting feature. Currently, the focus of DPsim is more on the application in distributed real-time co-simulation than power electronics. The idea is that the larger the simulation step, the smaller the impact of the communication delay between simulators, which are geographically distributed.

Distributed real-time co-simulation is motivated by large system simulations requiring more simulation capacity than is locally available and by the possibility of Hardware-In-the-Loop (HIL) testing with hardware under test and simulators at different locations. Using dynamic phasors for this application is a fairly recent development although the dynamic phasor, or the envelope function concept, is well known and was introduced in power electronics analysis in [1] as a generalized state space averaging method.

The authors of [2] have developed a distributed real-time simulation laboratory by applying a communication platform as a simulator-to-simulator interface in order to enable remote and online monitoring of interconnected transmission and distribution systems. Each simulator carries out simulations in time domain, while the variables exchanged at the interconnected nodes, i.e. decoupling point, are in the form of time-varying Fourier coefficients. The electromagnetic transient (EMT) values cannot be exchanged at every simulation step due to the communication delay. This delay may be directly incorporated into the physical power system model using traveling wave transmission line models [3]. However, electromagnetic waves travel about 15 km in 50 μs, a time which equals the typical step time in real-time EMT power system simulation. This means that a communication delay in the range of milliseconds would have to be compensated with a line length of several hundred kilometers which may require large and unrealistic changes to the original system model. Therefore, this method is suited for applications on local simulation clusters where the delay is only few simulation steps, but not for internet-distributed simulation, where the expected delay is often tens of milliseconds. In the latter case, the insertion of a transmission line model with the required parameters into the model would have a severe impact on the behavior of the system. Without compensation, the communication delay might cause large errors and even instability of the simulation as shown in [4] for a delay of more than 30 ms.

Starting from the concept presented in [2], we propose interfaces and system-wide simulation state variables based on dynamic phasors. This approach has two benefits explained in [5], which presents a comparison of DP and EMT simulations conducted in DPsim. First, it takes advantage of the computational efficiency of dynamic phasors in scenarios that involve bandpass signals with large center frequencies, as in the case of switching power electronics. Secondly, it increases the simulation time step thus reducing the difference between the local simulation time step size and the round trip time between simulators.

The approach employed in [2] requires the extraction of phasor information from the EMT signals. Therefore, the transparency of the interface is not guaranteed since the interface algorithm may alter the exchanged signals. This extraction step is eliminated by using dynamic phasors as state variables. Because of these features, DPsim is now part of the distributed real-time co-simulation platform described in [6], which is also the base for the experiment described in [2]. A first example of distributed real-time co-simulation using DPsim is presented in [7]. This example shows the impact of the latency in geographical distributed simulation on the results and how the latency can be modeled a priori to running actual simulations.

Besides, there are other relevant open-source initiatives in the field of power system simulation to be mentioned. In particular the Modelica community is very active in adapting Modelica environments for large scale power system cases [8,9] and providing comprehensive libraries of models [10,11]. These initiatives aim to enable large scale, fast simulation of power systems using open-source components but the focus is slightly different compared to the work presented here. The primary objective of DPsim is to assure a deterministic time step in terms of simulated and computation time to provide real-time capability required for Hardware-in-the-Loop (HIL) experiments. This is why in DPsim higher order solver methods are avoided and the system is split into subsystems, which are solved separately. The aforementioned related work does not seem to rely on such compromises since a deterministic time step is not required. While Modelica allows a convenient definition of physical models, DPsim does not intend to provide a solution in this regard. The idea is rather to extend the set of available models in DPsim by generating C-code from Modelica models to represent components connected to the network. These can be linked to DPsim and solved by the ODE solver that was integrated into DPsim for this purpose.

Another related work which is not open-source but also DP based is described in [12]. Similarly to DPsim, the authors have developed a simulator based on the DP approach. However, the focus does not seem to be fast simulation or real-time simulation since the simulator was developed in Python and there are no measures described in order to speed up the simulation. Furthermore, the validation is focusing on the correctness of the simulation rather than simulation speed.

2. Software description

The main theoretical building blocks of DPsim are the dynamic phasor concept and the modified nodal analysis (MNA). Dynamic phasors [13] have various names in scientific literature. Depending on the field and application, they are known as generalized averaging method [1], shifted frequency analysis [14], equivalent envelope [15,16] or baseband signal [17]. Here, the term dynamic phasors is selected because it is widely known in the power system community. The basic idea of dynamic phasors is to approximate a time domain signal x with a Fourier series representation as shown in (1)

$$x(\tau) = \sum_{k} X_k(t) e^{jkw_s(\tau)} \tag{1}$$

where $\tau \in (t - T, t]$. The kth coefficient is determined by

$$X_k(t) = \langle x \rangle_k(t) = \frac{1}{T} \int_{t-T}^{t} x(\tau) e^{-jkw_s(\tau)} d\tau \tag{2}$$

where ω_s is the fundamental system frequency and $k\omega_s$ are its harmonics.

MNA is used for the representation of the network as a linear equation system, whereas complex components, such as electrical machines, connected to the network are treated by a separate ODE solver. Depending on the simulation scenario, the admittance matrices of the required network topologies can be preprocessed, i.e. factorized, before simulation start to guarantee a deterministic simulation time step.

The simulation kernel of DPsim is extended with interfaces to support different use cases such as circuit or system simulation, batch simulation, co-simulation and HIL testing. DPsim supports the Common Information Model (CIM) [18] as native input for the description of electrical network topologies, component parameters and load flow data, which is used for initialization. Users can interact with the simulation kernel via Python bindings, which can be used to script the execution, schedule events, change parameters and retrieve results. Python scripts have been proven an easy and flexible way to codify the complete workflow of a

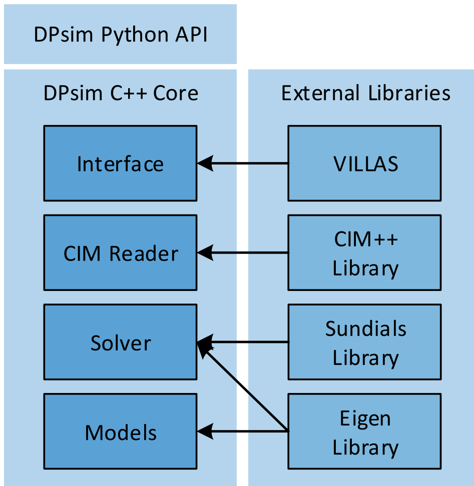


Fig. 1. Overview of DPsim's main components and dependencies on external libraries.

simulation from modeling to analysis and plotting, for example in Jupyter notebooks using Numpy and Matplotlib. Furthermore, DPsim supports co-simulation and interfaces to a variety of communication protocols of commercial hardware via the integration of DPsim with the *VILLASframework* [19], that enables large-scale co-simulations and HIL experiments.

2.1. Software architecture

2.1.1. Module structure

The DPsim simulation kernel and its component models are implemented in the C++ programming language. The availability of good compilers and highly optimized software libraries, such as *Eigen* [20], *Sundials* [21] and *VILLASnode*, which is part of the VILLASframework, for C++ were key factors for this decision.

The core of DPsim consists of models and solvers as depicted in Fig. 1. Interfaces to other software, hardware or data are supported through external libraries. Grid data in CIM standard format is imported using the *CIM++* library [22]. Communication with other software, e.g. real-time simulators, control and monitoring software, as well as hardware is provided by the VILLASframework. For linear algebra operations, DPsim uses the Eigen library. The MNA solver of DPsim uses the Eigen LU factorization and *Eigen::Dense/Sparse::Matrix* are the standard data types for all matrix variables. The Sundials solver package is included in DPsim to provide more complex ODE solvers for components connected to the network.

Compilation from source code requires a Git, a C++11 compiler and CMake 3.6. Tested compilers are Clang, GCC, MSVC and Intel's ICC. As a base operating system Ubuntu 18.04 LTS or Fedora 29 are recommended. For real-time execution a Linux 4.9 kernel with the *PREEMPT_RT* patch-set is recommended.

2.1.2. Class hierarchy

Simulation is the main interface class for users to control the simulation state. The attributes of a *Simulation* instance hold all the information that specifies one simulation scenario in DPsim. The *Simulation* holds references to the solvers, interfaces and the power system model information. This hierarchy is presented in Fig. 2. The complete class hierarchy diagram can be found in the developer's documentation of DPsim [23]. All solvers inherit from *Solver*, e.g. *MNASolver* and *ODESolver*, and all interfaces from *Interface*, e.g. for VILLASnode. All component models, power system, signal etc., derive from the class *IdentifiedObject*. The main attribute of this class is a unique identifier, which is equivalent to the *mRID* in CIM.

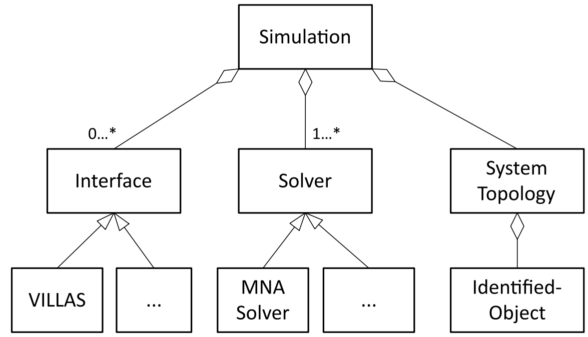


Fig. 2. Diagram of the main classes of DPsim.

2.1.3. Parallelization

In order to utilize the multiple processor cores of modern computer systems and speed up the simulation, the computation of one time step is split up into multiple tasks. These tasks are defined by the various parts of the simulation (like instances of *Solver* and *Interface*). An internal scheduler creates a task graph by analyzing which variables are modified and used by the tasks. This task graph is a directed acyclic graph with nodes representing tasks and edges representing data dependencies. The scheduler uses this graph to distribute the tasks onto multiple threads such that these dependencies are upheld. DPsim implements different scheduling algorithms for this purpose, the simplest of which being level scheduling as used in [24]. This algorithm divides the tasks into ordered levels such that each task only depends on tasks in a previous level. To simulate a time step, these levels are then executed in order by distributing all tasks of one level evenly among the available threads.

To further optimize the simulation performance for large networks when using multiple processors, a decoupling transmission line model can be introduced. In this model a transmission line is represented using equivalent current sources and resistances at the line ends that are not topologically connected as described in [25, ch. 6.2]. Instead, the ends are indirectly connected by updating the values of the sources based on the values on the other end only after a delay τ, which depends on the line's parameters. As subsystems that are connected using this line model are not connected in a strict topological sense, they can be solved independently and in parallel. This significantly reduces the computational effort to simulate large systems.

2.2. Software functionalities

DPsim supports both dynamic phasor and EMT simulation. Furthermore, DPsim is optimized for real-time simulation but it is also possible to run the same simulation scenario offline meaning that the simulation is executed as fast as possible. These different simulation modes are compatible with the user and simulation interface which are covered in the following.

2.2.1. User interface

Network models can be directly defined in the C++ code. This technique is complemented by a CIM importer, which allows the user to directly load network models from CIM-XML files. This proves to be an adequate form to describe network topology and component parameters, which are required by the solver. This CIM importer relieves the user from defining the model in plain C++ code. However, CIM-XML is not suitable for the definition of more complex simulation scenarios with time varying parameters or topology changes caused by contingencies in the system (e.g. breaker events or faults). Therefore, DPsim features Python bindings to most parts of the C++ programming interface. A scripting language like Python is used to define scenarios by leveraging the flexibility of a general purpose imperative language. This allows the user to write a single Python script to:

- Describe the network topology and parameters
- Load a network from CIM data and optionally extend it
- Define a simulation scenario with events or parameter changes
- Execute the simulation and analyze or plot the simulation results

2.2.2. Simulation interface

Interfacing the simulation kernel is desirable for multiple reasons:

- Real-time exchange of simulation signals for co-simulation or HIL testing
- User interface for online monitoring and control of the simulation
- Logging of simulation results for offline analysis
- Import of time series data, for example load and production profiles

For commercial simulation tools these interfaces are a major selling point as new protocols and standards and DAQ cards are introduced continuously. Their implementation and maintenance is time consuming and seemingly never ending. By design, DPsim tries to avoid this pitfall by leaving the implementation of interfaces, data formats and protocols to a separate project. VILLASnode, a component of the VILLASframework project, handles input/output as well as translation between different protocols. DPsim focuses on solving the system model and provides only a single type of interface, shared memory, to the VILLASnode gateway. Interfaces to external systems, databases, files or the web interface are then handled by the wide range of supported interfaces of the VILLASnode gateway, which in this case acts as a proxy between the shared-memory interface to DPsim and the outside world. Responsibilities are clearly separated. This allows the development of new interfaces without having to modify the simulation kernel itself.

In addition, DPsim can use the VILLASnode interfaces for co-simulation with other simulators or remote DPsim instances. In such a scenario, DPsim is usually coupled using an Ideal Transformer Model (ITM). Fig. 3 shows a decoupled model, which exchanges voltages in one and current signals in the opposite direction to control respective sources. For a phasor simulation, the exchanged signals are complex-valued attributes which are passed via the shared-memory interface to VILLASnode which further sends them to a remote simulator using one of VILLAS' supported protocols (e.g. MQTT, UDP, ZeroMQ, …). For geographically distributed simulations, VILLASnode can implement interface algorithms to compensate for the inherent communication latencies when executed in real-time over a high latency connection such as the internet. Alternatively, the implementation of a Discrete Fourier Transform (DFT) in VILLASnode allows for a coupling of the phasor-based DPsim with other EMT-based simulation tools like OPAL-RT or RTDS.

DPsim exchanges simulation data with the VILLASnode gateway via a shared-memory region. The execution of DPsim and VILLASnode as independent processes is crucial in real-time simulation scenarios as the main simulation kernel must not be interrupted by background activities such as data logging to a possibly blocking database.

Furthermore, DPsim has its own simple logging module to write results to CSV files for archival and post processing. This method is easy to setup and convenient for small simulations where post processing and analysis of simulation results is done in MATLAB or Python. In the long term, the internal CSV logging functionality is planned to be incorporated into VILLASnode and enhanced by the support of additional data formats like HDF5 as used by MATLAB.

3. Implementation and empirical results

To complement the previous overview of DPsim's architecture, the next subsection explains implementation specifics affecting the real-time performance of DPsim. The real-time performance of DPsim is demonstrated in the following subsection. The remaining two subsections demonstrate the correctness of the solution computed by DPsim against Matlab Simulink. The first simulation validates only the network solution, which is computed by the MNA solver. The second simulation features a combination of the MNA solver for the network solution and an ODE solver for the numerical integration of the synchronous generator equations.

3.1. Implementation details

Only a compiled language like C++ with minimal runtime overhead is suitable for the implementation since DPsim is targeting simulation time steps in the range of milliseconds to microseconds on off the shelf computing hardware. Great care was taken to avoid memory allocation during the actual simulation. Whenever possible, DPsim utilizes low order integration methods and avoids iterative solver strategies to minimize computation time. That is why the network part is handled by the MNA solver specifically developed for DPsim.

DPsim is compatible with Windows, macOS and Linux operating systems. Eventually, the operating system configuration can have a large impact on the real-time performance. To guarantee real-time execution, DPsim leverages several Linux real-time features such as the real-time capable *SCHED_FIFO* scheduler, real-time signals, the *timerfd* interface, or control groups (*cgroups*). Many of these features are nowadays incorporated in the standard Linux kernel but have their origin in the *PREEMPT_RT* patch-set. The patch-set is slowly integrated into the mainline kernel but still exists and can be applied to further improve the real-time performance by enabling preemption of critical sections such as interrupt handlers. The capability to preempt critical sections in the operating system kernel reduces the overall system latency and therefore helps to ensure strict deadlines at each time-step interval.

Real-time execution on Windows or macOS is not supported. Best real-time performance was achieved on a recent Intel x86_64 multi-core machine with optimized BIOS settings to avoid interruptions of the system by the System Management Mode (SMM). To do so, DPsim execution threads are isolated from remaining system processes using Linux's *cgroup* feature. This reduces the impact of background jobs in the system on the real-time performance.

As a good start for real-time optimizations, we recommend Redhat's Real-time Guide [26] with its *tuned* tool and the *realtime* profile. An updated list of optimization options can be found in the DPsim documentation [23].

To control the time step, DPsim is using *timerfd* interface in Linux environments. The *timerfd* interface allows for the configuration repetitive interval and one-off timers. It uses blocking file descriptors to suspend the execution of the simulation loop until the beginning of the next interval. This approach is more

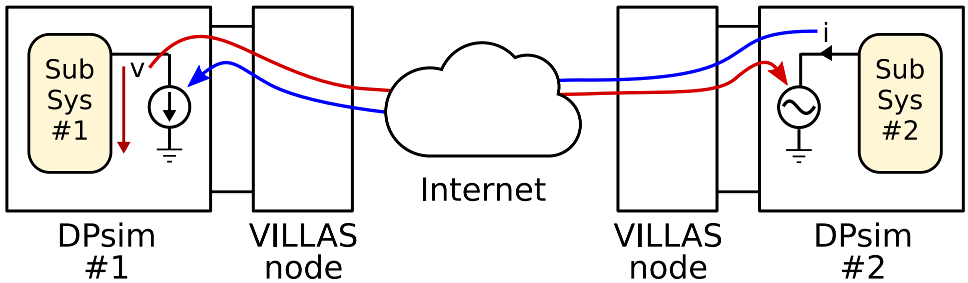


Fig. 3. VILLASnode as a gateway for distributed co-simulation with DPsim.

efficient than the use of the more common *timer* interface, which relies on signals. At the same time, *timerfd* is more accurate as calls to *sleep* or *nanosleep* as they suffer of a lingering drift to non-equidistant execution intervals. The *timerfd* is also used to schedule the synchronized start of distributed simulations as described in the introduction.

Shared-memory is a common method for inter-process communication (IPC) used on symmetric multi-processing (SMP) machines. It enables user processes to exchange data without the involvement of the OS kernel. Shared-memory IPC allows both processes, the solver and the gateway, to be executed in parallel while streaming their data with minimal latency over a queue in the shared memory region. The queue is implemented as a lock free multiple producer/multiple consumer (MPMC) queue and relies on atomic operations of the processor to facilitate synchronization of the DPsim and VILLASnode processes. The lockfree queue is a thread safe data structure. It is used to pass samples of simulation data between the involved processes. As such *message passing* is used as the main paradigm to avoid data races. During the initialization phase, semaphores are used to avoid race conditions in the setup of the shared-memory regions. The absence of the operating system in the communication is crucial to avoid costly context switches, which have to be avoided in a real-time context. Using the shared-memory interface, the DPsim simulation loop can run uninterrupted in a high priority process. At the same time, VILLASnode gets assigned a lower priority for handling of possibly blocking disk or database accesses. In a hardware-in-the-loop simulation it might be necessary to have hard real-time capable interfaces to the real world. For this purpose, DPsim supports an arbitrary number of shared-memory interfaces at the same time. This allows the user to configure one VILLASnode process with a high priority for the control of PCIe FPGA or DAQ cards, and at the same time another VILLASnode process for low priority logging of simulation data in the background.

3.2. Real-time performance evaluation

DPsim is specifically designed for real-time simulation. To assess the effect of system size on the real-time performance of DPsim, a simple test network was copied multiple times and connected with additional transmission lines. Fig. 4a shows the WSCC 9-bus system, which was used for this purpose. The copies were connected in a ring-like topology using additional transmission lines at the nodes 5, 6, and 8. The average wall clock time needed to simulate one time step was measured for a simulation time period of 0.1 s with a time step of 100 µs. Each measurement was further averaged over ten simulations of the same system. The measurements were performed on a system running Ubuntu 16.04 on an Intel Xeon Silver 4114 processor featuring ten cores clocked at 2.2 GHz. The results are shown in Fig. 4b for two different configurations: For the normal simulation, the additional transmission lines were modeled using the Pi model and only one thread was used. For the parallel simulation, the decoupling transmission line model was used for the additional lines and ten threads were employed. As it can be seen, the use of multiple threads and the special transmission line model greatly reduces the wall clock time necessary for simulating a single time step. Even for 40 copies of the original system (resulting in a system size of 360 nodes), the wall clock time per step stays under the simulation time step of 100 µs, thus allowing for real-time simulation. For a small number of system copies, DPsim supports simulation time steps of about 10 µs, which is a typical time step for commercial EMT simulators. However, it should be noted that such small time steps are not the aim of DPsim since the simulation is conducted in DP and not EMT.

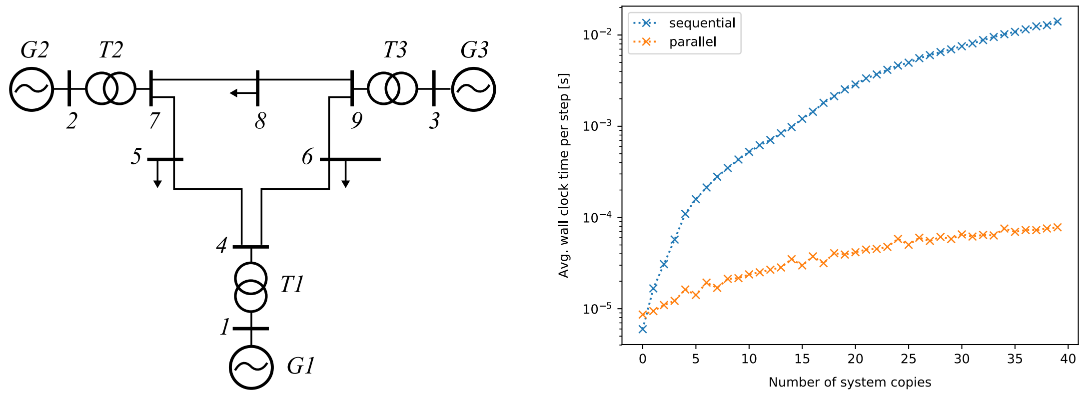
3.3. Validation of the MNA network solver

The next simulation case demonstrates the accuracy of the MNA network solver compared to Simulink results. Both simulators DPsim and Simulink are run with a simulation time step of 100 µs. It can be seen that for such small time steps DP simulations yield the same results as EMT. The larger the time step, the more will the results degrade. It is shown in [5] that the degradation of the results with larger time steps is smaller when using the DP approach compared to EMT.

Fig. 5a shows the simulated circuit, which is composed of one current source of 10 A, two resistors of 1 Ω, an inductor of 1 mH and a capacitor of 1 mF. The voltage source is set to its nonzero peak value at the beginning because it is following a cosine with zero phase shift. Therefore, the system is not starting from steady-state and a transient can be observed. The DPsim dynamic phasor results are transformed to time domain values and compared against Simulink EMT results. As can be seen from Fig. 5b, the results match.

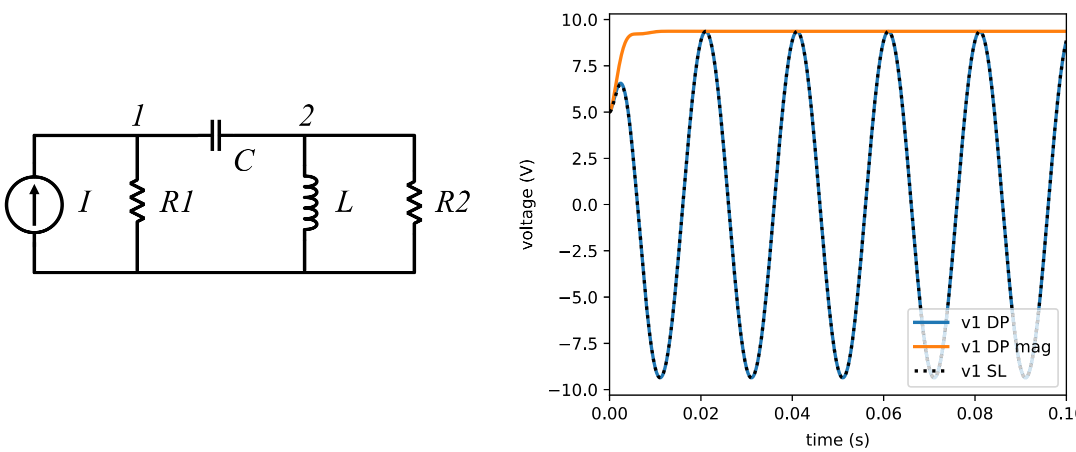
3.4. Validation of the ODE solver for components

The following example compares the results of Simulink and DPsim for a three phase synchronous generator fault. Here, the simulation time step is 50 µs for DPsim and Simulink. As in the previous simulation case, the DP approach would allow for larger time steps than EMT. A comprehensive study investigating this property and featuring synchronous generator models is presented in [27]. Initially, the load is 300 MW and the fault is applied at 0.1 s. The generator parameters are taken from example 3.1 in [28]. As in the previous example, the dynamic phasor results are transformed to time domain values before the comparison. Again, it is visible that the DPsim results match the Simulink results except when the fault is cleared (see Fig. 6). In contrast to DPsim, the fault in Simulink is not immediately cleared but at the next zero-crossing.



(a) (b)

Fig. 4. WSCC 9-bus system (a) and average wall clock time per step (b).



(a) (b)

Fig. 5. Example circuit (a) and DPsim dynamic phasor and Simulink EMT simulation results for node 1 (b).

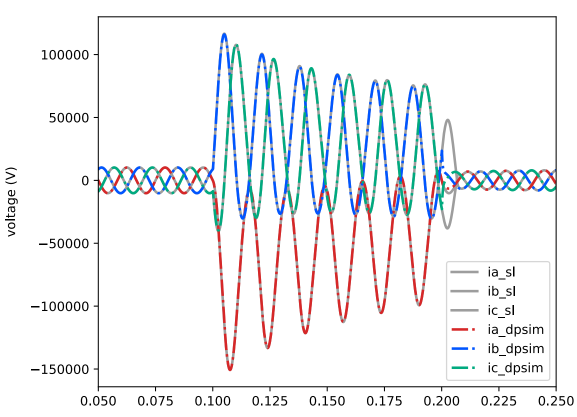


Fig. 6. DPsim dynamic phasor and Simulink EMT results for the synchronous generator three-phase fault example.

4. Illustrative examples

As mentioned in Section 2.2, there are two ways to define a circuit topology for DPsim: coding the topology using Python or C++ or importing it from CIM. The two options are presented by means of two examples: a circuit and a small power system. The first example presented in this section demonstrates the definition utilizing Python while the second example takes advantage of the CIM import function.

4.1. Defining a circuit simulation in python

The circuit of the previous section, Fig. 5, is taken as an example to demonstrate how circuits can be defined using DPsim's Python interface. The topology can be created in Python as depicted by Listing 1.

Listing 1: Python code to define a circuit.

```
# Nodes
gnd = dpsim.dp.Node.GND()
n1 = dpsim.dp.Node('n1')
n2 = dpsim.dp.Node('n2')

# Components
cs = dpsim.dp.ph1.CurrentSource('cs')
cs.I_ref = complex(10,0)
r1 = dpsim.dp.ph1.Resistor('r_1')
r1.R = 1
c1 = dpsim.dp.ph1.Capacitor('c_1')
c1.C = 0.001
l1 = dpsim.dp.ph1.Inductor('l_1')
l1.L = 0.001
r2 = dpsim.dp.ph1.Resistor('r_2')
r2.R = 1
```

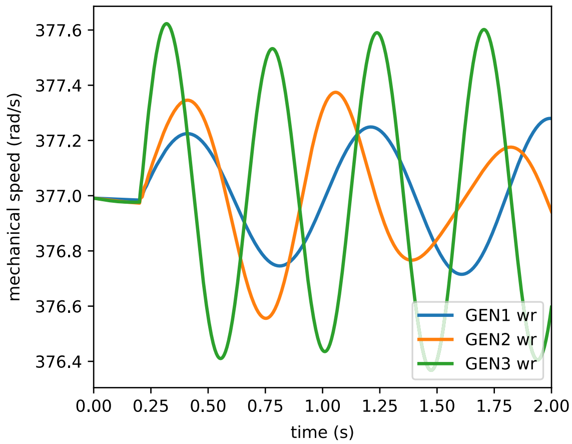


Fig. 7. WSCC 9-bus system mechanical speed simulation results after fault.

```
# Connections
cs.connect([gnd, n1])
r1.connect([n1, gnd])
c1.connect([n1, n2]);
l1.connect([n2, gnd]);
r2.connect([n2, gnd]);

system = dpsim.SystemTopology(50, [gnd, n1, n2], [cs, r1, c1, l1, r2]);
sim = dpsim.Simulation('circuit', system, timestep=0.0001, duration=0.1)
sim.start()
```

First, the nodes and components are declared and parameterized. Then, the connections between components and nodes are set and in the following step all network objects are added to the system topology. Finally, the system topology and parameters such as time step and final time can be used to create a simulation instance, which can be started, stopped and stepped through. Optionally, initial voltages could be assigned to the nodes. Since this is not the case here, the initial voltages are set to zero.

4.2. Simulating a power system defined in CIM

The next example presents the CIM loading functionality, which is used to read the data of the WSCC 9-bus system as displayed in Fig. 4a. The objects defined in the CIM file, e.g. *Terminal*, *TopologicalNode*, *SynchronousMachine*, are mapped to DPsim objects according to the *CIM::Reader* class of DPsim. The system frequency, 60 Hz, is not defined in the CIM file. Therefore, it needs to be specified before loading the CIM file. Furthermore, a fault is applied to node 9 of the imported system. To implement the fault, the system is extended by a switch connected to node 9, which connects the node to ground with a small resistance after 0.2 s. The switching action is created as an object of type *Event* and added to the *Simulation* instance.

Listing 2: Python code to import a topology from CIM.

```
# Read from CIM
files = glob.glob(' ../../ dpsim/Examples/CIM/WSCC09_RX_Dyn/*.xml')
system = dpsim.load_cim('WSCC9bus', files, frequency=60)

# Get existing nodes
gnd = dpsim.dp.Node.GND()
bus9 = system.nodes['BUS6']

# Add switch
sw = dpsim.dp.ph1.Switch('Switch')
sw.R_open = 1e9
sw.R_closed = 0.1
sw.is_closed = False
sw.connect([ bus9, gnd ])
system.add_component(sw)

sim = dpsim.Simulation('WSCC9bus', system,
timestep=0.0001, duration=2, init_steady_state=True)

sim.add_event(0.2, sw, 'is_closed', True)
sim.start()
```

In Fig. 7, it can be seen how the rotational speed of the generators starts to oscillate after the fault. The oscillation frequency depends on the mechanical inertia and as expected the generator with the largest inertia has the lowest oscillation frequency.

5. Impact

Since DPsim is open source, it can serve as a reference implementation for real-time power system simulators and a common basis for users working with different real-time simulation solutions. Having a common basis facilitates discussions on differences in simulation results of different tools. Besides, DPsim allows for real-time simulation on standard computing hardware, making real-time applications available to a wider range of researchers.

Thanks to its design, DPsim facilitates distributed real-time co-simulation, which promotes collaboration and allows the use of the simulation capacity of geographically distributed laboratories to support large scale scenarios [5]. Distributed real-time co-simulation allows also for better data privacy, enabling cooperation because, in a co-simulation, each entity can keep its data confidential and only exchange boundary variables. This could be interesting for confidential grid data but also black box device models where manufacturers cannot share implementation details.

The dynamic phasor approach is used here in a system level simulation in contrast to component level power electronics applications as in [1]. With an increasing share of power electronics in power systems, this approach supports the investigation of future grids.

DPsim is already used in the EU H2020 research project RE-SERVE [29] and it was developed as a solution to decrease the difference between communication delay and simulation time step in previous co-simulation projects, e.g. RT-Superlab [2]. DPsim is promoted by the FEIN association that also hosts its code and documentation [23].

6. Conclusions

The presented software project, DPsim, exploits the dynamic phasor approach to overcome the requirement for EMT simulation that the simulation time step needs to be proportional to the highest signal frequency. In doing so, DPsim facilitates distributed real-time simulation and lets users exploit simulation resources in different geographical locations. For this purpose, DPsim is programmed in the C++ language and has its own MNA based network solver. Despite having a C++ core, the DPsim allows for scripting simulations via the python interface and reading grid topologies in the standard CIM format via the CIM++ library. These features are demonstrated in two simulation examples, a circuit and a grid simulation. Likewise, the computational correctness of DPsim and its real-time performance are demonstrated by means of simulation examples. Furthermore, DPsim is tightly integrated with the VILLASframework that offers many interfaces to commercial real-time simulators and hardware.

Declaration of competing interest

We confirm that there are no known conflicts of interest associated with this publication.

Acknowledgments

This work was partly funded by the European Unions Horizon 2020 Framework Programme for Research and Innovation under grant agreement no 727481.

References

[1] Sanders SR, Noworolski JM, Liu XZ, Verghese GC. Generalized averaging method for power conversion circuits. IEEE Trans Power Electron 1991;6(2):251–9.
[2] Monti A, Stevic M, Vogel S, De Doncker RW, Bompard E, Estebsari A, Profumo F, Hovsapian R, Mohanpurkar M, Flicker JD, et al. A global real-time superlab: enabling high penetration of power electronics in the electric grid. IEEE Power Electr Mag. 2018;5(3):35–44.
[3] Schutt-Ainé JE. Latency insertion method (LIM) for the fast transient simulation of large networks. IEEE Trans Circuits Syst I 2001;48(1):81–9.
[4] Stevic M, Monti A, Benigni A. Development of a simulator-to-simulator interface for geographically distributed simulation of power systems in real time. In: Industrial electronics society, IECON 2015-41st annual conference of the IEEE. IEEE; 2015, p. 005020–5.
[5] Mirz M, Estebsari A, Arrigo F, Bompard E, Monti A. Dynamic phasors to enable distributed real-time simulation. In: Clean electrical power (ICCEP), 2017 6th international conference on. IEEE; 2017, p. 139–44.
[6] Vogel S, Mirz M, Razik L, Monti A. An open solution for next-generation real-time power system simulation. In: Energy internet and energy system integration (EI2), 2017 IEEE conference on. IEEE; 2017, p. 1–6.
[7] Mirz M, Vogel S, Monti A. First interconnection test of the nodes in pan-european simulation platform. RESERVE Library; 2017.
[8] Braun W, Casella F, Bachmann B, et al. Solving large-scale modelica models: new approaches and experimental results using openmodelica. In: 12 international modelica conference. Linkoping University Electronic Press; 2017, p. 557–63.
[9] Guironnet A, Saugier M, Petitrenaud S, Xavier F, Panciatici P. Towards an open-source solution using modelica for time-domain simulation of power systems. In: 2018 IEEE PES innovative smart grid technologies conference Europe. IEEE; 2018, p. 1–6.
[10] Casella F, Leva A, Bartolini A. Simulation of large grids in openmodelica: reflections and perspectives. In: Proceedings of the 12th international modelica conference, vol. 132. Linköping University Electronic Press; 2017, p. 227–33.
[11] Baudette M, Castro M, Rabuzin T, Lavenius J, Bogodorova T, Vanfretti L. OpenIPSL: Open-Instance power system library—Update 1.5 to "iTesla Power Systems Library (iPSL): A modelica library for phasor time-domain simulations". SoftwareX 2018;7:34–6.
[12] Martí AT, Jatskevich J. Transient stability analysis using shifted frequency analysis (SFA). In: 2018 power systems computation conference. IEEE; 2018, p. 1–7.
[13] Demiray T, Andersson G, Busarello L. Evaluation study for the simulation of power system transients using dynamic phasor models. In: Transmission and distribution conference and exposition: Latin America, 2008 IEEE/PES. IEEE; 2008, p. 1–6.
[14] Martí JR, Dommel HW, Bonatto BD, Barrete AF. Shifted frequency analysis (SFA) concepts for EMTP modelling and simulation of power system dynamics. In: Power systems computation conference. IEEE; 2014, p. 1–8.
[15] Strunz K, Shintaku R, Gao F. Frequency-adaptive network modeling for integrative simulation of natural and envelope waveforms in power systems and circuits. IEEE Trans Circuits Syst I Regul Pap 2006;53(12):2788–803.
[16] Suárez A. Analysis and design of autonomous microwave circuits, vol. 190. John Wiley & Sons; 2009.
[17] Proakis JG. Digital communications. New York: McGraw-Hill; 1995.
[18] Energy management system application program interface (EMS-API) - Part 301: Common information model (CIM) base. International Electrotechnical Commission; 2016.
[19] FEIN Aachen e. V., VILLAS framework. http://www.fein-aachen.org/projects/villas-framework. [Accessed 23 November 2018].
[20] Guennebaud G, Jacob B, et al. Eigen v3. 2010, http://eigen.tuxfamily.org.
[21] Hindmarsh AC, Brown PN, Grant KE, Lee SL, Serban R, Shumaker DE, Woodward CS. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. ACM Trans Math Softw 2005;31(3):363–96.
[22] Razik L, Mirz M, Knibbe D, Lankes S, Monti A. Automated deserializer generation from CIM ontologies: CIM++ - an easy-to-use and automated adaptable open-source library for object deserialization in C++ from documents based on user-specified UML models following the Common Information Model (CIM) standards for the energy sector. Comput Sci Res Dev 2018;33(1–2):93–103.
[23] FEIN Aachen e. V., DPsim. https://www.fein-aachen.org/projects/dpsim/. [Accessed 23 November 2018].
[24] Walther M, Waurich V, Schubert C, Gubsch I. Equation based parallelization of modelica models. In: Proceedings of the 10th international modelica conference. p. 1213–20.
[25] Watson N, Arrillaga J. Power systems electromagnetic transients simulation. IET; 2003.
[26] Red hat enterprise Linux for Real Time 7: Tuning Guide; 2018. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux_for_real_time/7/pdf/tuning_guide/Red_Hat_Enterprise_Linux_for_Real_Time-7-Tuning_Guide-en-US.pdf.
[27] Zhang P, Marti JR, Dommel HW. Synchronous machine modeling based on shifted frequency analysis. IEEE Trans Power Syst 2007;22(3):1139–47.
[28] Kundur P, Balu NJ, Lauby MG. Power system stability and control, vol. 7. New York: McGraw-hill; 1994.
[29] RESERVE. http://www.re-serve.eu/. [Accessed 23 November 2018].

DOI: 10.1109/ICCEP.2017.8004805

Publisher version: https://ieeexplore.ieee.org/document/8004805

Dynamic Phasors to Enable Distributed Real-Time Simulation

Markus Mirz*, Abouzar Estebsari†, Francesco Arrigo†, Ettore Bompard†, Antonello Monti*

*Institute for Automation of Complex Power Systems, RWTH Aachen University, Aachen, Germany
†Department of Energy, Politecnico di Torino, Turin, Italy

***Abstract*—Distributed real-time simulation allows the sharing of simulator equipment and components connected in Hardware-In-the-Loop experiments. In this paper, we analyze the challenges of geographically distributed real-time power system simulation and how dynamic phasors could be applied to improve the accuracy of the simulation results for large time steps. The time step is of particular interest since the communication delay between simulators interconnected through wide area network is much larger than the simulation time step typically used in electromagnetic transient real-time simulations. However, commercially available real-time simulators use either the electromagnetic transient or classic complex phasor representation. Results in the dynamic phasor and electromagnetic transient domain are compared to quantify the advantage of dynamic phasor simulations in practice. The test platform for this evaluation is a power system simulator which is currently under development.**

***Index Terms*—Power system simulation, Distributed computing, Real-time systems, Computational modeling, Power system modeling**

I. Introduction

Deployment of new power system components such as power electronics devices and measurement units, advanced communication infrastructure as well as new control and management algorithms is rapidly increasing. Therefore, it is important to study the interactions among these components to ensure the stability and controllability of future grids.

Real-time simulation is becoming increasingly popular as a means to test and validate physical components and algorithms in a controlled and realistic environment. The synchronization of simulation time with wall clock time allows the exchange of physical inputs and outputs between externally connected devices and the real-time simulator. When a device is attached to a simulation, so-called Hardware-In-the-Loop (HIL), the behavior of components in the system can be safely tested also during emergency operations, and the integration of the device into the system can be easily validated.

However, the real-time simulators and the devices-under-test might be geographically distributed or the capabilities of the local real-time simulator might not be sufficient for the given simulation scenario. Then, the model which is to be simulated can be partitioned for a distributed simulation. Typically, geographically distributed simulators are connected via Wide Area Networks (WAN). In addition to the challenge of partitioning the model and creating interfaces for parallel execution, the communication delay can be even larger than the simulation time step which is typically used in power system electromagnetic transient (EMT) simulations. One way to overcome this are interface algorithms that compensate for the delay [1].

In this paper, we pursue another strategy: increasing the simulation time step to enable the exchange of signals for each simulation step in real-time. The minimal frequency that has to be represented in power system simulations is the system frequency of $50\,\mathrm{Hz}$ or $60\,\mathrm{Hz}$ depending on the region. Hence, it would be advantageous to integrate the frequency implicitly as it is done in traditional complex phasor calculation but without being fixed to this frequency. Dynamic phasors meet both requirements [2]. Since commercially available real-time simulators do not provide dynamic phasor models, the advantage of dynamic phasor over EMT for distributed real-time simulations is not quantified, yet. That is why this paper presents a simulation study on EMT and dynamic phasor simulations for varying time steps. The simulator used in the study is currently developed at ACS, RWTH Aachen University.

The paper is structured as follows: Section II introduces the techniques and definitions which are used throughout the paper. Sections III and IV analyze the challenges of distributed real-time simulation in depth and define the scope of the solution presented here. The simulation study and results are given in Section V.

II. State of the Art

A. Real-Time Simulation

Simulations can be classified according to the relation between simulation time and wall clock time and the time flow mechanism, fixed / variable time stepped or event-driven simulation [3]. Typically, power system real-time simulators progress in time with fixed time steps since more sophisticated variable time step integration methods are not suitable for fast calculations as needed for real-time execution and small time steps. In case of nonlinear events such as transistor switching, the time step synchronization can give rise to jitter: events are considered only at the

This work was supported by the European Unions Horizon 2020 research and innovation programme under grant agreement No 727481.

beginning of the next step and not when they are actually happening. Particular techniques and solving methods are therefore needed to avoid numerical instability [4]. Real time simulation of power systems is intended to accurately reproduce the dynamic behavior of a physical system, for example, during very fast transient situations like system outage management or fault location [5]. Protection and control system development and testing, distributed generation modeling, especially with renewable energy resource (RES) integration, and microgrid control are some of the application fields in which real-time simulation is widely utilized [6].

B. Distributed and Parallelized Simulation

Parallel and distributed simulation may refer to sharing the computational load among several processing units which can belong to one computer, or multiple computers interconnected in a lab, or several computers located geographically distant.

In case of offline or as-fast-as-possible simulations where simulation time is not tied to wall clock time, the motivation for distributed computation is the acceleration of the simulation process by utilizing many processing units in parallel. In contrast, this paper is focused on parallel and distributed simulation of power systems in real-time. Again, one objective is to increase the overall computational power of the simulator. Besides, there are more advantages of distributed real-time simulation [7], [8]. The following list summarizes the most important ones:

- Available hardware and software in different real-time simulation laboratories can be shared among participants in order to enhance computation power and facilitate remote Software-In-the-Loop (SIL) and (Power-)Hardware-In-the-Loop
- Larger scales of systems could be simulated by assigning different parts of the model to several laboratories
- Expertise and knowledge in different energy fields and use cases could be shared by applying the same case study concurrently without the need to move researchers and equipment
- Confidential data does not need to be shared as each laboratory can be responsible for simulating its own part of the model locally, solely exchanging interface variables with other interconnected systems, imitating the real world where regional or national power grids are interconnected through tie-lines
- Several algorithms to control, manage, or regulate systems can be tested in laboratories where no realistic models of the environment (e.g. power grid model) are available

C. Dynamic Phasors

Dynamic Phasors are known as a powerful and efficient analytic tool used in simulations, which is based on the concept of time varying Fourier coefficients. As the size and number of new components of power systems are rapidly increasing, especially by introducing power electronics devices, there is a need for simulation methods which can address and describe the dynamic changes of grid states without demanding the same computational cost as EMT simulations. In this way, larger grids under several conditions can be simulated and studied.

Application of dynamic phasors in power systems is ubiquitous. It was firstly used for studying and simulating general converter technologies. Nowadays, one of the main goals is to simulate and integrate new Distributed Energy Resources (DER) and HVDC converter technologies. The aim is to construct an efficient model for the dynamics of switching gates phenomena with a high level of detail [2], [9], [10], [11].

Fault analysis and unbalanced conditions are other important research topics in which dynamic phasors allow larger models and more efficient simulations. Asymmetrical faults are studied in [12], [13], [14], [15]. In the last two articles the behavior of AC machines like Doubly-Fed Induction Machine (DFIG) wind turbines or synchronous generators are evaluated using dynamic phasors. In [16] an unbalanced distribution system consisting of a single-phase PV system, a three-phase induction machine and a three-phase power factor correction capacitor is simulated using dynamic phasors, trying to achieve a comprehensive modeling approach. Results show great similarity with time-domain simulations.

In [17] an effort to generalize the dynamic study with dynamic phasors is made by modeling and validating a multiple synchronous generator test grid. The goal was twofold: application of dynamic phasors for multi-source, multi-frequency systems and modeling of systems with time-varying frequencies. In [18] a frequency matched linear numerical integration technique is used to improve efficiency and accuracy of the dynamic phasor simulation.

Although dynamic phasors allow a significant saving in terms of computational cost, the idea to apply it to real-time simulation is still fairly novel. Commercial simulators like RTDS and OPAL-RT offer analytic modeling tools being able to perform EMT simulations e.g. eMEGAsim developed by OPAL-RT. OPAL-RT introduced also simulation tools in the traditional complex phasor domain called ePHASORsim which is limited to system fundamental frequency. As mentioned in Section I, dynamic phasors might allow larger systems to be simulated in real-time with larger time-steps (i.e. milliseconds instead of microseconds) while catching the dynamic behavior of a system with frequency deviation. In this paper, this capability is highlighted and proposed to be applied in distributed real-time simulations where communication latency and the size of the model lead to serious limitations when using conventional EMT solvers. The following section analyzes these limitations and challenges.

III. Challenges of distributed real-time simulation

Geographically distributed simulation borrows problems from parallel simulation. In both cases the model has to be partitioned and it has to be assured that the parallel simulation results do not differ from the results of a sequential simulation. Furthermore, real-time requirements decrease the number of suitable simulation techniques [19].

In case of geographically large distances between the simulators the time needed for information exchange can have a huge impact on the simulation. In fact, a communication delay of more than $10\,\mathrm{ms}$ might cause large errors and even instability [1]. One cause for this is the sampling requirement imposed if an AC $50\,\mathrm{Hz}$ or $60\,\mathrm{Hz}$ system is simulated in EMT. According to the sampling theorem, the minimum sampling frequency is twice the maximum frequency expected in the system. This combined with the large Round-Trip Time (RTT) expected in geographically distributed simulations, complicates the synchronization among the simulators. Simulations using traditional complex phasors do not impose the strong sampling requirement since the system frequency is implicitly included but this frequency is fixed. Therefore, this approach does not support frequency control or stability studies, for example, on transmission level.

The authors of [8] realized an integrated real-time co-simulation laboratory by applying a communication platform as a simulator-to-simulator interface proposed in [1] in order to enable remote and online monitoring of an interconnected transmission-distribution system. Based on that novel approach, each simulator carried out simulations in time domain, while the time-varying Fourier coefficients of the quantities in the interconnection node, i.e. decoupling point, are exchanged. EMT values could not be exchanged for every simulation step due to the communication RRT. Following transformed quantities could be exchanged: Traditional complex phasor or dynamic phasors of the fundamental and harmonic components. The problem with the former solution is the following. For transient analysis, frequency deviation in one side cannot be captured on the other side to perform a distributed real-time simulation with the same results as a local real-time simulation. With the latter solution, dynamic phasor exchange, simulations do not imply fixed system frequency. Still, the local EMT simulation is computational less efficient compared to phasor simulations. Besides, this approach requires the extraction of phasor information from the EMT signals. Therefore, transparency of the interface is not given since the interface algorithm may alter the exchanged signals. So far, the interface algorithm can extract magnitude and phase for several harmonic components. The frequency is assumed to be the nominal system frequency with a DC link connecting the two systems.

The following section describes the use case of our dynamic phasor solver which is supposed to avoid these limitations.

IV. Use case

Recently, real-time control of Distributed Energy Resources (DER), especially from renewable energy resources with intermittent behavior, is attracting more attention due to increasing penetration of such energy resources in grids. Renewable energy sources aim to reduce CO2 emissions, but as a side effect, power system stability and quality of electric power supply are jeopardized as they do not provide inertia by default due to power converters being used as interfaces to the grid.

A H2020 European project named RE-SERVE[1] was just initiated to address such challenges and investigate control strategies for DERs using a pan-European simulation network. In order to validate the performance of new control algorithms for large grid scenarios, integration of geographically distributed simulation facilities is anticipated in RE-SERVE.

The pan-European real-time simulation infrastructure will be implemented by interconnecting laboratory facilities of four universities, that are depicted in Figure 1, to test frequency and voltage control strategies, providing support to energy stakeholders and regulators in their decision making. Performing the simulations requires large computational power and efficient simulation solvers to enable large-scale network studies and validation of frequency and voltage control algorithms. Geographically distributed simulation would bring together available hardware facilities located in different laboratories.

As discussed in the challenges section, applying an alternative solver seems inevitable. Our dynamic phasor solver as an open source code together with the novel pan-European real-time simulation platform would enable an integrated European virtual simulation environment to simulate larger systems in real-time.

By integrating simulation facilities in different laboratories, the network data and system components do not need to be shared, and all susceptible data can be kept confidential. Only boundary quantities are exchanged via WAN interfaces. This approach would encourage system operators including DSOs and TSOs to perform simulations in collaboration with other laboratories without sharing confidential data. While keeping the main fundamental dynamics of the system, the proposed solver allows to increase simulation time-steps to enable large-scale power system simulation. Commercial EMT real-time simulators such as RTDS and OPAL-RT could be connected to this simulation to investigate specific parts of the grids with a small time step. This would require an interface algorithm between the EMT and the dynamic phasor part such as the one described in [8] but the interface between two dynamic

[1] http://www.re-serve.eu/

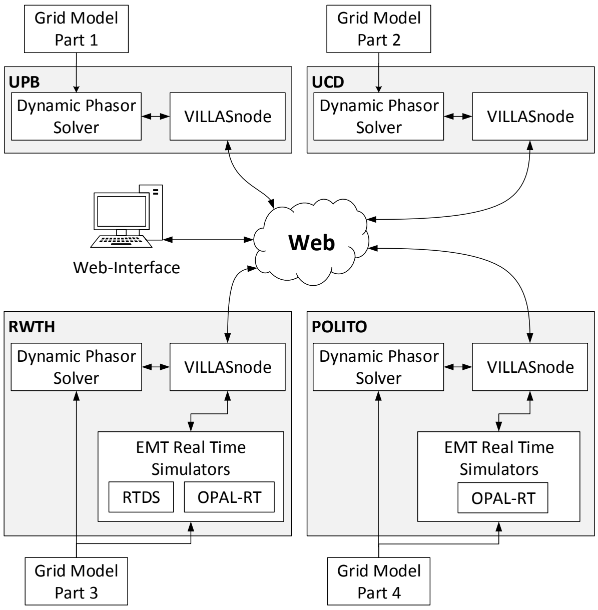


Fig. 1. Distributed real-time simulation infrastructure integrating laboratories of RWTH Aachen University, Politecnico di Torino (POLITO), University Politehnica of Bucharest (UPB), University College Dublin (UCD)

phasor simulators is facilitated a lot since the extraction of the phasor information is not required and the time step can be increased. Then, the communication delay is easier to compensate for.

V. Dynamic phasors in real-time simulation

As mentioned in Section III, the exchange of time-domain values among simulators imposes strong requirements on the sampling rate. Therefore, previous work already introduced dynamic phasors as a means of exchanging data in the frequency domain rather than the time domain [20]. However, this requires the extraction of the dynamic phasors from the time domain signal for every simulation step. Instead, we propose to simulate the entire system in dynamic phasors to be able to increase the simulation time step and to avoid the conversion from the time domain to the frequency domain.

A. Development of Dynamic Phasors

Dynamic phasors were initially developed for power electronics analysis [2]. Later, the concept was extended to power systems analysis [21]. The use of dynamic phasors for power system simulation is described in [22].

This section covers the general approach of dynamic phasors for power system simulation while pointing out the main features that are interesting for distributed real-time simulation. Using Dynamic phasors, it is possible to treat an AC signal as a DC signal without losing its dynamic properties as it is the case when using classic complex phasors in power system analysis. Instead of

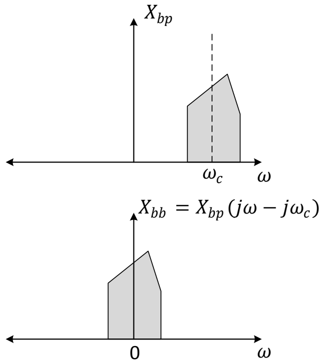


Fig. 2. Shift of band limited signal in the frequency domain

fixing the frequency, the signal is shifted by the system frequency, e.g. $50\,\mathrm{Hz}$. Besides, one time domain variable can be approximated by several dynamic phasors of different harmonics, each of these shifted by their center frequency. However, this shift only decreases the maximum frequency of the simulated signals if all frequencies of interest lie in a small band around these center frequencies. The fundamental frequency of power systems is normally varying in a region close to the nominal system frequency. Hence, the bandpass limitation is fulfilled. The shift in the frequency domain is visualized in Figure 2.

The bandpass signal X_{bp} centered around ω_c is real valued and can be represented in the right half plane of the frequency spectrum. The shifted signal, which is sometimes termed baseband signal, X_{bb} features a smaller maximum frequency. According to the sampling theorem, the baseband signal requires a smaller sampling rate to be represented correctly. This property is very important in the application of real-time simulation since the round trip time (RTT) between two simulators in different locations can be very significant. In case of pan-European simulations the RTT has been found to be several tens of ms [20], whereas links between Europe and the US can exhibit RTT of well over $100\,\mathrm{ms}$ [1]. Therefore, the default time step of $50\,\mu\mathrm{s}$, used by many commercial real-time simulators, does not allow a data exchange between the simulators for every simulation step without compensation for the communication delay.

In the following, the general dynamic phasor approach is explained which is the basis of the simulation example in the next subsection. First of all, the time domain signal x is approximated with a Fourier series representation.

$$x(\tau) = \sum_k X_k(t) e^{jkw_s(\tau)} \tag{1}$$

where $\tau \in (t-T, t]$. The k^{th} coefficient is determined by

$$X_k(t) = \langle x \rangle_k(t) = \frac{1}{T} \int_{t-T}^{t} x(\tau) e^{-jkw_s(\tau)} d\tau \tag{2}$$

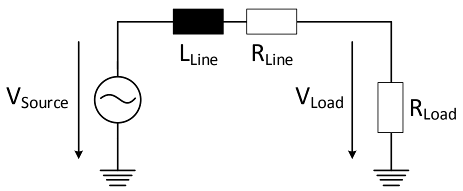


Fig. 3. Example circuit for the comparison of dynamic phasor and EMT simulations

where ω_s is the fundamental system frequency and $k\omega_s$ are its harmonics. Deriving equation (2) leads to

$$\frac{d}{dt}\langle x\rangle_k(t) = \left\langle \frac{d}{dt}x \right\rangle_k (t) - jk\omega_s\langle x\rangle_k(t) \tag{3}$$

Accordingly, a state space model of the general form

$$\frac{d}{dt}x(t) = f(x(t), u(t)) \tag{4}$$

would be transformed to

$$\frac{d}{dt}\langle x\rangle_k(t) = \langle f(x(t), u(t))\rangle_k - jk\omega_s\langle x\rangle_k(t) \tag{5}$$

Applying (5) to the equation of an inductance

$$\frac{d}{dt}i(t) = \frac{1}{L}\cdot v(t) \tag{6}$$

results in the following equation for the fundamental dynamic phasor:

$$\frac{d}{dt}\langle i\rangle_1(t) = \frac{1}{L}\cdot\langle v\rangle_1(t) - j\omega_s\langle i\rangle_1(t) \tag{7}$$

In the next section, a simple circuit, that includes an inductance modeled according to (6) and (7), is simulated using the EMT and dynamic phasor approach for different time steps.

B. Simulation Study for Different Time-Steps

To support the theoretical advantage of dynamic phasor over EMT simulations, we present the simulation results for a simple circuit as depicted in 3. The circuit consists of an AC voltage source of $V_{Source} = 1\,\mathrm{kV}$ peak voltage with a resistance of $R_{Source} = 1\,\Omega$, an RX-series element of $R_{Line} = 1\,\Omega$ and $L_{Line} = 100\,\mathrm{mH}$ and a load resistance of $R_{Load} = 100\,\Omega$. Internally, the voltage source is transformed to its Norton equivalent.

The simulation scenario is as follows. At $0.2\,\mathrm{s}$, the load resistance is decreased to $50\,\Omega$ and at $0.4\,\mathrm{s}$ the frequency of the AC voltage source is decreased from $50\,\mathrm{Hz}$ to $45\,\mathrm{Hz}$. This scenario is simulated for different time steps between $50\,\mathrm{\mu s}$ and $45\,\mathrm{ms}$ using our own simulator that is based on the resistive companion method. In the following, we compare the voltage V_{Load} across the load resistance.

As can be seen in Figure 4 and Table I, the results are almost identical for time steps of $50\,\mathrm{\mu s}$. Figure 4 shows the EMT results, the absolute value of the fundamental dynamic phasor and the time domain signal of the fundamental dynamic phasors after it is shifted back by $50\,\mathrm{Hz}$ in the frequency domain. The shift and transformation into the time domain is accomplished by taking the real part of the signal after applying equation (1). Furthermore, Table I depicts the mean squared error for the signals after linear interpolation. It is important to point out that the interpolation of the dynamic phasors is applied for real and imaginary part separately and before shifting the signal back to $50\,\mathrm{Hz}$. Comparing the $20\,\mathrm{ms}$ time step results presented in Figure 5 and 6, it can be seen that the dynamic phasor simulation is very accurate even for large time steps. Without interpolation, the fundamental sinusoidal is not represented correctly by the dynamic phasor values since the number of data points is too small.

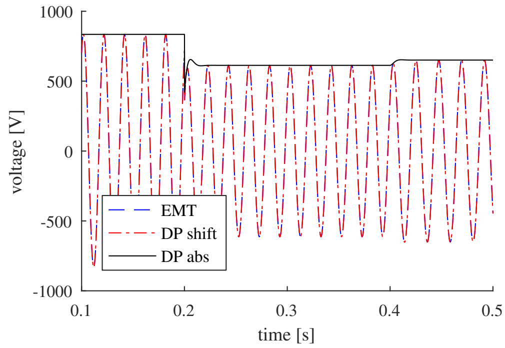


Fig. 4. Comparison of dynamic phasors and EMT simulation for time steps of $50\,\mathrm{\mu s}$

TABLE I
MEAN SQUARED ERRORS FOR V_{Load}

Timestep	EMT	EMT interp.	DP	DP interp.
50 µs	0	-	0.3973E−4	-
1 ms	105.6141	138.9220	97.0038	110.7485
5 ms	7.9126E+3	2.1133E+4	2.0867E+3	588.3851
10 ms	2.5305E+5	2.2442E+5	5.7648E+3	914.9802
15 ms	3.8408E+5	4.5783E+5	1.2080E+4	2.0503E+3
20 ms	2.4815E+5	1.0170E+6	1.6898E+4	3.1879E+3
25 ms	1.5840E+5	5.6950E+5	1.7379E+4	3.9371E+3
30 ms	1.4264E+5	3.1919E+5	2.6352E+4	3.2719E+3
35 ms	3.6699E+5	4.7725E+5	2.9534E+4	3.4567E+3
40 ms	2.6983E+5	0.9913E+6	2.8822E+4	3.6437E+3

From Table I, it can be concluded that the error is growing much slower for dynamic phasor simulations. Tens of ms seem to be feasible time steps for dynamic phasor simulations if the system transients are not too fast. Therefore, the dynamic phasor approach could enable distributed simulations without having to compensate for the communication delay in some cases, for example, distributed simulation among participants in Europe.

VI. CONCLUSION

In this paper, the requirements and challenges of distributed real-time simulation are deduced from use cases

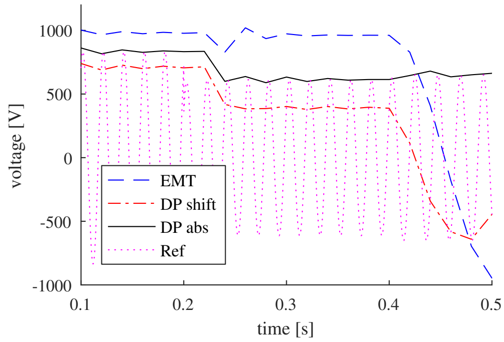


Fig. 5. Comparison of dynamic phasors and EMT simulation for time steps of 20 ms

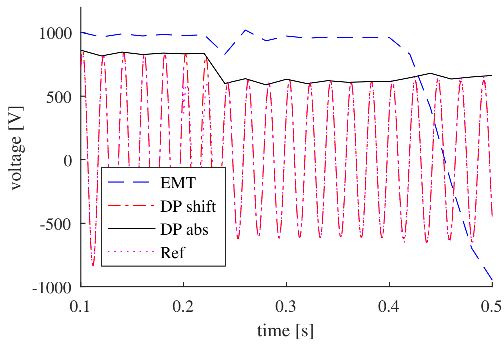


Fig. 6. Comparison of dynamic phasors and EMT simulation for time steps of 20 ms with interpolation

such as the pan-European real-time simulation of the power system. Distributed real-time simulation involving frequency stability and control are found to be very important. Dynamic phasors are proposed as solution to solve one of the main problems of distributed real-time simulation, the large round trip time between the simulators which determines the minimum time step if the simulators are to exchange data for every step. Furthermore, we present a study that shows the advantage of using dynamic phasors for distributed real-time simulation. The focus of this study is on the simulation time step. The larger the time step, the lesser the impact of the communication delay between two geographically distributed simulators on the real-time data exchange.

Currently, the real-time simulation capability is added to the dynamic phasor solver presented in Section V. As next step, we plan to investigate its capabilities regarding parallel power system simulation.

References

[1] M. Stevic, A. Monti, and A. Benigni, "Development of a simulator-to-simulator interface for geographically distributed simulation of power systems in real time," in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, Nov 2015, pp. 005 020–005 025.

[2] S. R. Sanders, J. M. Noworolski *et al.*, "Generalized averaging method for power conversion circuits," *IEEE Transactions on Power Electronics*, vol. 6, no. 2, pp. 251–259, 1991.

[3] R. M. Fujimoto, *Parallel and distributed simulation systems*. Wiley New York, 2000, vol. 300.

[4] J. Belanger, P. Venne, and J. Paquin, "The what, where and why of real-time simulation," *Planet RT*, vol. 1, no. 1, pp. 25–29, 2010.

[5] A. Estebsari, E. Pons *et al.*, "An iot realization in an interdepartmental real time simulation lab for distribution system control and management studies," in *Environment and Electrical Engineering (EEEIC), 2016 IEEE 16th International Conference on*. IEEE, 2016, pp. 1–6.

[6] X. Guillaud, M. O. Faruque *et al.*, "Applications of real-time simulation technologies in power and energy systems," *IEEE Power and Energy Technology Systems Journal*, vol. 2, no. 3, pp. 103–115, 2015.

[7] E. Bompard, A. Monti *et al.*, "A multi-site real-time co-simulation platform for the testing of control strategies of distributed storage and v2g in distribution networks," in *Power Electronics and Applications (EPE'16 ECCE Europe), 2016 18th European Conference on*. IEEE, 2016, pp. 1–9.

[8] M. Stevic, S. Vogel *et al.*, "Virtual integration of laboratories over long distance for real-time co-simulation of power systems," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Oct 2016, pp. 6717–6721.

[9] S. Chandrasekar and R. Gokaraju, "Dynamic phasor modeling of type 3 dfig wind generators (including ssci phenomenon) for short-circuit calculations," *IEEE Transactions on Power Delivery*, vol. 30, no. 2, pp. 887–897, 2015.

[10] A. Coronado-Mendoza, J. L. Bernal-Agustín, and J. A. Domínguez-Navarro, "Photovoltaic boost converter system with dynamic phasors modelling," *Electric Power Systems Research*, vol. 81, no. 9, pp. 1840–1848, 2011.

[11] H. Zhu, Z. Cai *et al.*, "Hybrid-model transient stability simulation using dynamic phasors based hvdc system model," *Electric power systems research*, vol. 76, no. 6, pp. 582–591, 2006.

[12] A. M. Stankovic and T. Aydin, "Analysis of asymmetrical faults in power systems using dynamic phasors," *IEEE Transactions on Power Systems*, vol. 15, no. 3, pp. 1062–1068, 2000.

[13] R. H. Salim and R. A. Ramos, "A model-based approach for small-signal stability assessment of unbalanced power systems," *IEEE Transactions on Power Systems*, vol. 27, no. 4, pp. 2006–2014, 2012.

[14] A. M. Stankovic, S. R. Sanders, and T. Aydin, "Dynamic phasors in modeling and analysis of unbalanced polyphase ac machines," *IEEE Transactions on Energy Conversion*, vol. 17, no. 1, pp. 107–113, 2002.

[15] T. Demiray, F. Milano, and G. Andersson, "Dynamic phasor modeling of the doubly-fed induction generator under unbalanced conditions," in *Power Tech, 2007 IEEE Lausanne*. IEEE, 2007, pp. 1049–1054.

[16] Z. Miao, L. Piyasinghe *et al.*, "Dynamic phasor-based modeling of unbalanced radial distribution systems," *IEEE Transactions on Power Systems*, vol. 30, no. 6, pp. 3102–3109, 2015.

[17] T. Yang, S. Bozhko *et al.*, "Dynamic phasor modeling of multi-generator variable frequency electrical power systems," *IEEE Transactions on Power Systems*, vol. 31, no. 1, pp. 563–571, 2016.

[18] T. Demiray and G. Andersson, "Optimization of numerical integration methods for the simulation of dynamic phasor models in power systems," *International Journal of Electrical Power & Energy Systems*, vol. 31, no. 9, pp. 512–521, 2009.

[19] A. Benigni and A. Monti, "A parallel approach to real-time simulation of power electronics systems," *IEEE Transactions on Power Electronics*, vol. 30, no. 9, pp. 5192–5206, 2015.

[20] M. Stevic, S. Vogel *et al.*, "Feasibility of geographically distributed real-time simulation of hvdc system interconnected with ac networks," in *2015 IEEE Eindhoven PowerTech*, June 2015, pp. 1–5.

[21] V. Venkatasubramanian, H. Schattler, and J. Zaborszky, "Fast time-varying phasor analysis in the balanced three-phase large electric power system," *IEEE Transactions on Automatic Control*, vol. 40, no. 11, pp. 1975–1982, 1995.

[22] T. Demiray, G. Andersson, and L. Busarello, "Evaluation study for the simulation of power system transients using dynamic phasor models," in *Transmission and Distribution Conference and Exposition: Latin America, 2008 IEEE/PES*. IEEE, 2008, pp. 1–6.