Fully Automated EMT Simulation of Multi-Terminal Direct Current Grids Using RE-INTEGRATE

Phani R V Marthi[1] marthip@ornl.gov
Phuong H. Hoang[1] hoangh@ornl.gov
Shamim Hasan[1] hasans2@ornl.gov
Suman Debnath[1] debnaths@ornl.gov

***Abstract*—In this paper, fully automated electromagnetic transient (EMT) simulation of multi-terminal direct current (MTdc) systems is presented for the first time. There has been research performed on alleviating the computational burden involved in EMT simulation of MTdc systems in the past. However, minimal research is conducted on fully automated EMT simulation of MTdc systems in commercial EMT software or free EMT software. Lack of fully automated EMT simulation of MTdc severely hinders the wider power system community (such as original equipment manufacturers (OEMs), regional transmission organizations (RTOs), among others) in performing essential studies needed to maintain large-scale power system reliability. Therefore in this work, several novel concepts pertaining to parsing, automation, and model scripting utilized specifically for end-to-end automated EMT simulation of MTdc systems are presented. To demonstrate the validity of these methods, a 3-terminal MTdc system is developed in EMT software RE-INTEGRATE.**

***Index Terms*—EMT, Automation, Parsing, MTdc, HVdc.**

I. Introduction

Multi-terminal direct current (MTdc) grids are and will be one of the essential commodities in the large-scale modern power grids integrated with power electronics (PE)-based systems. This is because of their well-known benefits such as efficient large power transfer, improved reliability, ease of PE-based generation and load integration, ease of maintenance, among others. Modeling and simulation of these massive structures was always a fascinating avenue of research, especially electromagnetic transient (EMT) simulation and modeling of MTdc systems.

In the prior research, there have been research on modeling of MTdc systems and fast EMT simulation of such systems for various scenarios and use cases [1], [2], [3]. Much of the focus was on model development for specific use-case, computational resource optimization and efficiency, and various control and stability method validation [4], [5]. Very less focus and attention was paid to the input data structures that can ease interoperability of multi-vendor simulations of MTdc systems across different software platforms and ease with scalability. Building on graphic user interface (GUI), which has been used traditionally, will have scalability challenges. Lack of standard data structures for MTdc systems and associated automated scalability are two major problems faced by the wider power system community in conducting large-scale interoperability, integration, and planning studies. These challenges severely impair the power system community in the model development, model interoperability, and model portability processes.

To address these challenges, novel concepts in the fields of parsing, automation, and model scripting are presented in this paper for fully automated EMT simulation of MTdc systems. The details regarding the novel concepts are: a) eXtensive Markup Language(XML)-based parsing is introduced for the first time as a potential standard practice for parsing the dynamic data of various components in the MTdc grids for EMT simulation; b) customized automation methods that align well with the hierarchical structure of xml is introduced; and c) legacy system level C++ language's class-based architecture is utilized as part of EMT software RE-INTEGRATE. The proposed novel methods are tested, validated, and demonstrated for the EMT simulation of 3-terminal MTdc system using EMT software RE-INTEGRATE.

II. MTdc GRID EMT Model Description

The overall 3-terminal MTdc system is showcased in Fig. 1. The fundamental components in the MTdc system are high-voltage direct current (HVdc) stations comprised of modular multi-level converters (MMCs) and MMC controllers, HVdc station controllers, transformers, transmission lines, and ac grid sources. In the case of this 3-terminal MTdc system, there is a high-voltage direct current (HVdc) stations at each terminal. Each HVdc station is modeled as a two terminal component where one end is connected to the alternating current (ac) grid source through a transformer and another end is connected to the three direct current (dc) $\pm 525\text{kV}$ transmission lines with neutral.

In the configuration showcased in Fig. 1, the HVdc station is considered to be an symmetric bi-pole with the optional metallic return dc cable. Each station consists of two MMCs, an upper MMC and a lower MMC. Each MMC consists of six arms, and each arm consists of an arm inductor, arm resistor,

Research sponsored by the U.S. Department of Energy. The views expressed herein do not necessarily represent the views of the U.S. Department of Energy or the United States Government. [1]Energy Science & Technology Science Directorate, Oak Ridge National Laboratory, Knoxville, TN, USA.

979-8-3315-8138-1

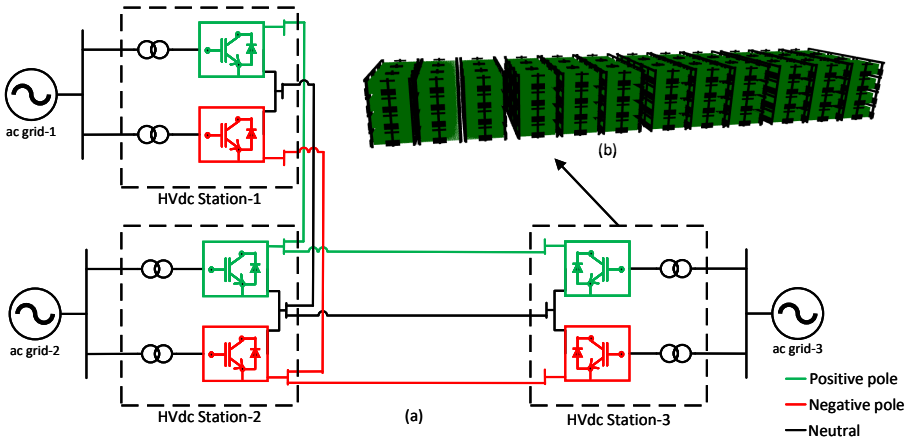

Fig. 1. (a) 3-terminal MTdc Grid Layout; (b) 3D visualization of a HVdc station

and N half-bridge submodules. The MMCs within this bi-pole station are modeled as detailed switching models employing hybrid discretization methods described in [3].

For brevity, further details about the modeling methods, simulation algorithms employed, and blocking conditions are not discussed in this paper and are described in detail in [3]. These dynamics and their corresponding differential states are exclusively associated with the MMCs in the HVdc station. Apart from these differential states, there are several algebraic states that are exclusively associated with the voltage states introduced in the model to measure the node voltages(to be precise there are total of 34 algebraic states between HVdc station, its internal components such as MMCs and other components in the MTdc grid such as transformers).

The transformer model utilized in the 3-terminal MTdc system is a delta-star ground transformer. The dynamic models of ac transformers are modeled based on the classical approach i.e., equivalent circuit of two mutually coupled windings model. The ac sources are modeled as a 3-phase voltage sources. However, the ac-source can be replaced with a more detailed ac grid model that consists of synchronous generators, exciters, governors, transmission lines, transformers, and loads. The dc transmission line dynamic models can be modeled based on the frequency dependent transmission line models as detailed in [6] or can be modeled based on the cascaded pi-line approach detailed in [7].

III. XML Parsing Architectures

XML parsing architecture consists of three main components: a) well-formed XML file; b) XML parser; and c) XML schema[1]. In this section, the prime focus is on the first two components of XML parsing architectures. In the first component, i.e., well-formed XML file, all the MTdc grid component's data is encoded in the nested group structure. An example of the nested well-formed XML file for MTdc grid is shown in listing 1.

[1]There are no mandatory rules to utilize the schema for XML parsing. As long as XML file is well-formed, the XML parser should be able to parse the complete data in the XML file.

Listing 1. Example well-formed XML for MTdc Grid

```
<MTdc>
   <StationGroup>
     <Station>
         <name></name>
         <value></value>
              ...
         <Transformer>
            <name></name>
            <value></value>
            ...
         </Transformer>
         ...
         <MMC_hardware>
            <name></name>
            <value></value>
            ...
         </MMC_hardware>
         <MMC_software>
            <name></name>
            <value></value>
            ...
         </MMC_software>
       ...
         <Station_hardware>
            <name></name>
            <value></value>
            ...
         <Station_hardware>
     </Station>
     ...
   </StationGroup>
   <ac_sourceGroup>
     <ac_source>
         <name></name>
         <value></value>
          ...
     </ac_source>
     ...
   </ac_sourceGroup>
   <dc_lineGroup>
     <pi_line>
         <name></name>
         <value></value>
              ...
     </pi_line>
  ...
   </dc_lineGroup>
</MTDC>
```

In the nested structure shown in listing 1, the root element *<MTDC>* consists of multiple groups such as *<StationGroup>*, *<ac_sourceGroup>*, and *<dc_sourceGroup>*. Each group inherently has subgroups such as *<Transformers>*, *<MMC_hardware>*, *<MMC_software>*, among others. The components data is encoded in the individual elements of these subgroups as showcased in listing 1. The parsing file utilizes the Uniform Resource Identifier (URI) reference "http://www.w3.org/2001/XMLSchema-instance". Further information on URI can be found in detail in [8].

The next important aspect of the XML parsing architecture is the parser. The XML parser is developed in C++ and utilizes the functionalities of the extremely fast and lightweight *RapidXML* XML library header file. The pseudo code for the XML parser called as *xml_parser.cpp* is showcased in listing 2.

In the *xml_parser.cpp*, functions are created for different groups and smart pointers for the subgroups such as *<MMC_hardware>*, and *<MMC_software>*. Smart pointers are utilized to ensure no manual assignment of memory is done and more importantly memory does not get corrupted. The elements are created using the data structure XML_ELEM.

The benefits of the proposed XML parsing architecture are multi fold. Some of them are: a) the data can be vendor agnostic; and b) facilitates easier model portability and interoperability of different vendor-based MTdc grid systems. These advantages enable the proposed XML architecture as a potential practise for standard parsing of MTdc grid systems.

Listing 2. Pseudo code for xmlparser.cpp

```
// data structures for elements
struct XML_ELEM
{
   // name of the node
   // name of the element
   // value of the element
};
class XMLParser{
   private:
      // dynamic array of elements
      // smart pointers for all components
      // functions for different groups
      // smart pointers for sub groups
   public:
      XMLParser(); //constructor
      ~XMLParser(); //destructor
      // filepath assignment
      // functions to pass data to all groups
          and sub groups
};
```

IV. Automation

A. Overall Process

In this section, the behind the scenes automation processes for the EMT simulation in software RE-INTEGRATE are discussed. For the simulation of the EMT simulation of 3-terminal MTdc grid (or any other large-scale power grid with PE-based systems), user needs to provide only a single file: EMT project simulation settings file. In this file, all simulation settings associated with EMT simulation of the 3-terminal MTdc system such as simulation time-step, solver

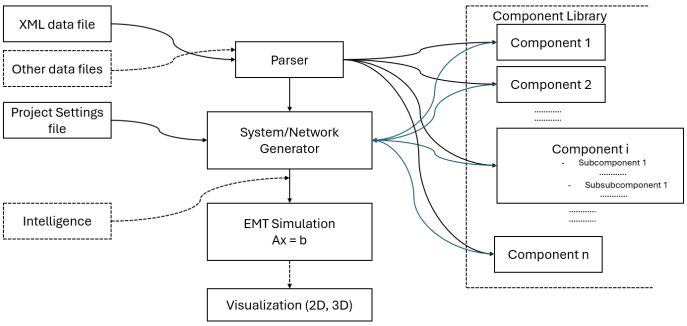

Fig. 2. Overall framework of the RE-INTEGRATE EMT software

selection, well-formed XML file path, and other attributes are specified. As the step one in automation, a class object for the EMT_simulation is created that takes in the project settings information and then passes the information to the system generator. Based on the information passed to the system generator, creation of objects associated with each component and data initialization of each component from the parsed XML file are completed. Based on the bus and node information from the parser, connectivity of different components is identified and appropriately system network of 3-terminal MTdc grid is created. The system network is created based on Kirchoff's voltage law (KVL) and Kirchoff's current law (KCL). Another important aspect in this process is to reduce the redundant states that are shared by two components connected to each other, for example the node voltage shared between two connecting components. Once the network is created, then the next step is selection of the appropriate solver[2] to solve the network based on the information provided in the project settings file.

B. Framework Description

The overall framework of the RE-INTEGRATE EMT software is illustrated in Fig. 2. The framework for the RE-INTEGRATE consists of the following vital modules: a) project settings; b) data files; c) parser; d) system/network generator; d) components library; e) intelligence; f) Ax = b for EMT simulation; and g) visualization. The settings needed for the EMT simulation are listed in the project settings module. The path for the various data files are also included here along with other settings such as solver selection, time-step, and simulation time, among others. In RE-INTEGRATE code-base, the data file module has the capability to use different data files such as .raw, .dyr, and .dss files apart from .XML file. The parser module can read data from these multiple data files and can pass the data to different modules in RE-INTEGRATE code-base. Based on the data parsed from the parser, the network is generated in the system/network generator module. In the process of developing the network, the system/network generator module and parser module interact with another important module i.e. component library module. Component library module consists of the component model scripts for

[2]There are currently many solvers that users can select in RE-INTEGRATE EMT software. Examples include: dense solver, KLU, BD (MPI and CUDA versions), and BBD (MPI and CUDA versions). The RE-INTEGRATE software also has the capability to facilitate external solvers that users want to utilize.

various power system components that are typically observed in the network. These component scripts consists of the linearized component models' A matrices and b vectors of different power system components' linear system of equations Ax = b that are generated based on the nodal approach. The data parsed from the parser is relayed to the parameters section in the components models (parameters can be seen in the generic class structure of the HVdc station listed in listing 3) and the parameters within the component are aptly utilized to generate the A matrix and b vector (the functions that generate A matrix and b vector can also be observed in the listing 3). The generated different components' A matrices and b vectors are then sent to the system/network generator module and it is here they are stitched together to form entire network A matrix and b vector. Additionally, system/network generator module has automated processes that take care of the redundant states briefly mentioned in Section IV-A to ensure no numerical instabilities occur.

Once the entire system A matrix and b vector are generated, they are simulated at every time-step using the EMT simulation Ax = b module. In this module, the list of solvers, and processes required for solution of Ax = b linear system of equations at every time-step are present. The EMT simulation solution at every time-step (or multiple time-steps based on user's input) is then sent to the visualization module. In the visualization block the real-time 3D rendering of the model along with component model dynamics are included. The 3D rendering of the HVdc station is shown in Fig. 1b. There is an additional important module in the RE-INTEGRATE code-base, i.e., intelligence module (highlighted in dashed lines). Intelligence module is invoked typically between the system/network generator module and the EMT simulation Ax = b module. In the intelligence module, there are advanced re-ordering schemes that re-order the system A matrix and b vector to aid in faster simulation. In addition to the advanced re-ordering schemes, there are detection algorithms that specify which solvers should be utilized. The intelligence module is not utilized in the proposed work. However, these functionalities currently exist in the RE-INTEGRATE and will be integrated in this use case in future.

V. Component Scripting

The entire scripting of the RE-INTEGRATE code-base is done in C++ to leverage the advantage of the object oriented programming concepts such as classes [9]. These concepts are extremely useful in developing very complex hierarchical and modular projects such as EMT simulation of 3-terminal MTdc grids. The C++ classes developed for the 3-terminal MTdc system are *StationMMC.cpp*, *HardwareMMC.cpp*, *SoftwareMMC.cpp*, *Transformer.cpp*, *dcline.cpp*, and *acsource.cpp*. From the hierarchy point of view, *StationMMC.cpp* is the top most layer i.e., component layer, the next layer is *HardwareMMC.cpp* which is the subcomponent layer, and finally followed by the *SoftwareMMC.cpp* which is the subsubcomponent layer. This hierarchical layer of the components can be pictorially visualized in component i of Fig. 2. The generic class structure of the HVdc station is listed in Listing 3.

Listing 3. Pseudo code for the HVdc station

```
class StationMMC : public generic component
       , public generic subcomponent
{
public:
      // constructor for parameter
          initialization
     StationMMC(const STATION_PARAM& params);
      // constructor
     StationMMC(A matrix,
       parameters,
       mapping information,
       b vector states,
       x vector states,
       subcomponent data structures);
      // A matrix generation
     void Generate_Acomp();
      // b vector generation
     void Generate_Bcomp();
     // updates at each simulation timestep
     virtual void timestep_update();
     // destructor
     ~StationMMC();
      // parameter initialization
     void CompInitialization();
private:
      // declaration of states
      // declaration of internal variables
      // declaration of external variables
      // declaration of b vector
      // global offsets for component's A
          matrix
      // subcomponent function generator
      void generate_subcomponent(struct
          subcomp, int index);
};
```

The most important elements each PE-based components' code *StationMMC.cpp* are component constructor *StationMMC*, component A matrix generator *Generate_Acomp*, component b vector generator *Generate_Bcomp*, component's subcomponent generator *generate_subcomponent*, and component updater at every time-step *timestep_update*. The essential functionalities of each element are: a) in the constructor, all the parsed data is passed on to the parameters vector *parameters*, A matrix, and b vector are generated for component with all the subcomponents information; b) the entire component's A matrix in the sparse coordinate (COO) format is embedded in the *Generate_Acomp* function; c) the b vector information is embedded in the *Generate_Bcomp* function; d) the exchange of the information within components, subcomponents, and subsubcomponents, updates to the subcomponent's at each time-step, are some of the very vital tasks performed within the *timestep_update*; and e) the parameter allocation, subcomponent matrix allocation within the component matrix based on the column map generated for the component are the tasks performed within the *generate_subcomponent*. For brevity, only extremely crucial tasks and functionalities of the components scripts are discussed in this section.

The pivotal innovation in the RE-INTEGRATE's component scripting architectures lies in the successful realization of the underlying complex hierarchy associated with the different components of the 3-terminal MTdc system. In the case of 3-terminal MTdc grid EMT simulation, the proposed scripting methods are able to pass the parameters

of various components such as <Station>, <Transformer>, <MMC_hardware>, <MMC_software>, <ac_source>, and <pi_line> given in XML data file (as shown in Listing. 1) to the component classes *StationMMC.cpp*, *Transformer.cpp*, *HardwareMMC.cpp*, *SoftwareMMC.cpp*, *acsource.cpp*, and *dcline.cpp* respectively.

VI. Case Studies and Simulation Results

A. Case Study: 3-terminal MTdc System

The case-study to demonstrate the validity of the proposed innovative methods for a 3-terminal MTdc grid shown in Fig. 1. In this MTdc system, HVdc station models from different architectures are developed that mimic the architectures from different vendors. The two station models are based on the symmetric bi-pole configuration with metallic return. Each HVdc station has two MMC converters and their associated control systems.

The control architectures include station control algorithms such as droop control, converter control algorithms such as ac current control, circulating current control, and capacitor voltage balancing control algorithms. The transformers utilized are the delta-star ground transformers. The dc transmission lines are rated at $\pm 525\text{kV}$. The dc transmission line models are developed based on the pi-line configuration. In this case-study, there are total 3 terminals, 3 HVdc stations, 6 MMC converters, 6 transformers, 3 dc transmission lines and 3 ac grid sources.

B. Simulation Results

The simulation results for the EMT simulation of the 3-terminal MTdc system using RE-INTEGRATE are shown in Fig. 3-Fig. 5. From the simulation results, it is observed that the overall EMT simulation of 3-terminal MTdc grid using RE-INTEGRATE was successful with the input data defined in XML file for different station architectures. The arm currents for one of the HVdc terminal station's lower MMC converter are shown in Fig. 3. From the figure, it can be clearly observed that the arm currents are accurate and stable. Similarly from the Figs. 4-5, it can be clearly observed that the ac-side power profiles of all the MMC converters, and dc grid voltages are accurate and stable. The key take-away from this simulation is that the proposed methods have the capability to simulate different vendor models using a defined data structure and hierarchical model architecture that helps with scalability. Thus demonstrating the ease of interoperability and enabling wider power system community to perform system studies without any hindrance.

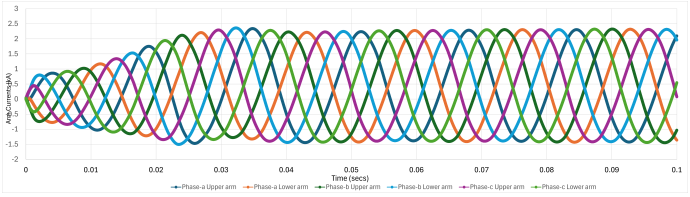

Fig. 3. One of the HVdc station terminal's lower MMC arm currents

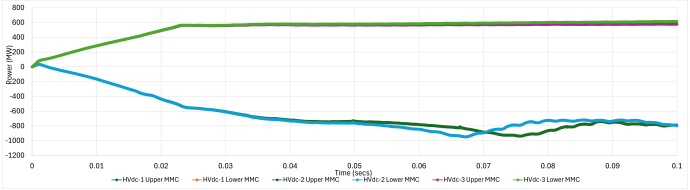

Fig. 4. 3-terminal MTdc grid power profiles

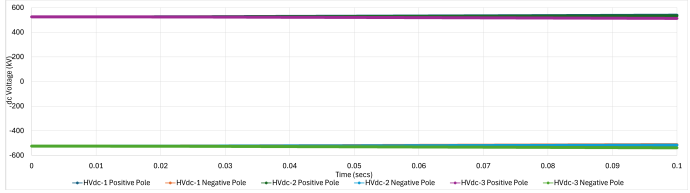

Fig. 5. 3-terminal MTdc grid dc grid voltage profiles

VII. Conclusion

In this paper, novel concepts pertaining to EMT simulation of MTdc grids using EMT simulation software RE-INTEGRATE are proposed. This concept utilizes hierarchical data and model structures that are helpful for multi-vendor implementations and scalability. The proposed methods were tested on a 3-terminal MTdc grid use-case. From the simulation results, it was observed that the proposed methods were successful in performing the EMT simulation of the 3-terminal MTdc system. The successful end-to-end EMT simulation for such complex hierarchical MTdc systems using EMT software RE-INTEGRATE is reported for the first time.

The success of the EMT simulation case study in this research work is a big step towards creating a fully functional EMT simulation software RE-INTEGRATE. With this proposed framework, minimal user intervention is needed with data structures handling most of the information along with the automation scripts developed. In future, computational performance of the different solvers for MTdc systems will be evaluated along with intelligent re-ordering schemes.

References

[1] D. Shu, X. Xie, Q. Jiang, G. Guo, and K. Wang, "A multirate emt co-simulation of large ac and mmc-based mtdc systems," *IEEE Transactions on Power Systems*, 2018.

[2] A. Allabadi, J. Mahseredjian, K. Jacobs, S. Dennetière, I. Kocar, and T. Ould-Bachir, "Initializing large-scale multi-terminal hvdc systems using decoupling interface," *IEEE Transactions on Power Delivery*, 2024.

[3] S. Debnath and M. Chinthavali, "Numerical-stiffness-based simulation of mixed transmission systems," *IEEE Transactions on Industrial Electronics*, 2018.

[4] N. Lin and V. Dinavahi, "Parallel high-fidelity electromagnetic transient simulation of large-scale multi-terminal dc grids," in *2020 IEEE Power Energy Society General Meeting (PESGM)*, 2020.

[5] N. R. Chaudhuri and B. Chaudhuri, "Adaptive droop control for effective power sharing in multi-terminal dc (mtdc) grids," in *2013 IEEE Power Energy Society General Meeting*, 2013.

[6] J. Sun, S. Debnath, M. Saeedifard, and P. R. Marthi, "Real-time electromagnetic transient simulation of multi-terminal hvdc–ac grids based on gpu," *IEEE Transactions on Industrial Electronics*, 2021.

[7] N. R. Chaudhuri, B. Chaudhuri, R. Majumder, and A. Yazdani, *Multi-Terminal Direct-Current Grids: Modeling, Analysis, and Control*. Hoboken, NJ: Wiley-IEEE Press, 2014.

[8] H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn, "XML schema part 1: Structures, second edition," World Wide Web Consortium (W3C), W3C Recommendation REC-xmlschema-1-20041028, 2004, last visited: ¡insert date here¿. [Online]. Available: https://www.w3.org/TR/xmlschema-1/

[9] S. Debnath, J. Choi, H. Hughes, K. Kurte, P. Marthi, and S. Hahn, "High-performance computing based emt simulation of large pv or hybrid pv plants," in *2023 IEEE Power Energy Society General Meeting (PESGM)*, 2023.