ATPMAT: An Open Source Toolbox for Systematic Creation Of EMTP Cases in ATP Using Matlab

Ahmad Abdullah

Texas A&M University

Department of Electrical and Computer Engineering

College Station, TX, 77840

E-mail: ahmad@tamu.edu

*Abstract*—This paper introduces ATPMAT which is an open source Matlab-based toolbox to automate many tasks done in ATP faced by researchers and engineers. The toolbox provides a rich set of M-files that is GPL based to change ATP files and run ATP from within Matlab. The toolbox provides a systematic way not only to change one ATP file but to batch process ATP files to create a batch of faults, line switching or lightning strike cases for any line. The heart of the toolbox is a data translator that interprets the contents of the ATP file and inserts appropriate lines of code to accommodate changes requested by the user. The toolbox then makes changes to ATP file according to user input and then run ATP from within Matlab. The output of the toolbox is a set of mat files each of which contains the voltages and currents form both ends of the line under study. The mat files is saved in a folder the use specified by the user.

Keywords— ATP, EMTP, Data Translator, MATLAB, Open Source Toolbox, Faults, Lightning, Line Switching.

# Introduction

Generating hundreds or even thousands of simulations using ATP have become such a common task in most academic institutions, yet there is no free tool publicly available to generate such simulations. The power system committee has seen an increase in application of artificial neural networks to power system problems notably in classification ‎[1]. ANN and other methods have been successfully applied to classification of power quality disturbances ‎[2]. Also transient based protection schemes have been proposed in literature ‎[3]. In TBP schemes and all classification problems, a large data set has to exist for the evaluation of the method proposed. In case of fault classification ‎[4], a large number of simulations has to be carried out in order to verify the method. In case of ANN ‎[5] a large number of cases has to exist for ANN training, validation and testing. Even though such studies need a lot of cases for verification, yet there is no systematic way for creating such cases for research purposes. In ‎[6] a Matlab algorithm is given for automatic creation of fault cases without releasing any source code.

In this paper, we provide ATPMAT®: a Matlab® toolbox to use ATP ‎[7] in patch processing mode; that is to create many simulations with one click of a button after the user supplies parameters of simulations he wants to generate. The code that is freely available on a git repository ‎[8] is used to automate creation of fault cases, line switching cases, line opening, and lightning strike simulations. It is assumed that the user has a network that is already built in ATP and he wants to make changes to it to create a batch of simulations. Thus the toolbox we are sharing is not used to create ATP files but to modify existing file to accomplish the studies mentioned above.

The user has three functions to choose from: CreateFault, SwitchLine, and LightStrike. The CreateFault function creates a fault on a line to be chosen by the user. The SwitchLine switches the line on or off based on the time the user specifies. The LightStrike function strikes a line determined by the user by a lightning strike with certain amplitude at certain time. The user supplies the above aforementioned functions with certain parameters for the case to run. If the user wishes to create a batch of simulations, of faults for example, then he needs to include such function in a for loop. Such a function has been also supplied and it is called BatchProcessATP and can be considered the central point of the toolbox. The toolbox provides a rich set of M-files to change ATP files. For example: the user can insert a switch at the beginning or the end of the line and set its opening and closing times; the user can specify the magnitude and the rise/fall time of the lightning surge and its location; the user can select measuring nodes as well. The user also has the option to convert PL4 files to mat files using PL42mat ‎[9] utility. The user needs to adjust the settings of ATP before using the toolbox.

The code has been tested on Windows 7 running on an Intel core i7 processor. The paper builds upon a paper that we published before in IPST 2011 ‎[10] which the user need to consult in case he needs to see how the toolbox is used to create a real case for relay testing. This is the first version of the toolbox and we will be increasing its capabilities over the course of next versions. We also share ATP model for IEEE 118 ‎[11] case as a test system in our online git repo ‎[8]. The modeled is shared mainly to provide an online tutorial on how to use the toolbox.

The paper is organized as follows: Section II discusses how to set up system environment for Windows. Section III gives an overview of the structure of the toolbox and the functions supported by it. Section IV contains the conclusion and future work.

# Setting Up Sysmtem environment

Before using the toolbox, the user has to prepare her system for doing so. The user has to obtain the following files: runAF.exe, runATP.exe, makeATP.exe, PL42mat.exe and mytpbig.exe. All of those files have to be put on the same folder. The user will need to keep the path for this folder as this folder will be passed to the function BatchProcessATP. It should be noted that all the above files can be obtained when the user first agree to the terms and conditions of the ATP license ‎[12]. Before using the toolbox, the user has to adjust listsize.big. The easiest way to adjust listsize.big is to open ATP Launcher then click on make Tpbig.exe from the tools menu. All the user needs to do is just enter the values given in a file uploaded to the bitbucket repository ‎[8] and then click on make. Besides obtaining the above files, the user must create a directory for his output files. The will need to enter the URI of this directory to BatchProcessATP. It is in this folder the user will find all output mat files of his simulations. This completes the setting up of environment in Windows.

# Overview of Toolbox

The overall toolbox structure is shown the flow chart in fig. 1. The starting function of the toolbox is CreateScenarios function. This is the function that’s responsible for translating user input into a data structure that can be understood by the toolbox. After creating that structure, the user invokes the function BatchProcessATP which is the central point of the toolbox. To use BatchProcessATP, the user supplies the file path of the ATP file, the directory of all ATP related executables obtained in Section II, a user specified directory for the output of toolbox, the type of study and the structure that as created using CreateScenarios.

The type of the study has to be one of three options: “fault”, “lighting” or “energisation”. The information in the structure will vary according the study sought. If the user wishes to create a batch of faults, he needs to supply the following to CreateScenarios function: fault distance, fault resistance, fault type, line under consideration, start time, incipient angle and end time. If the user wants to create a lightning case, he needs to supply the line under consideration, the lightning amplitude, the phase being hit, as well as the distance to be hit. If the user wishes to energize a line, he needs to specify the line under consideration, which terminal to be energized and the angle at which the line will be energized. The inner working of the functions CreateFault, LightStrike, and SwitchLine will be described in the next subsections in more detail but in this section we focus on the overall picture and the systematic way of creating EMTP cases.

A fault or a lightning strike includes having the line divided into two sections. In case of fault, we create a fault at the common point of both sections. In case of lightning we insert a lightning source at the common point of both sections. In those two cases, we need to divide the line into two sections and insert a block- to be explained in next sections- in the common point to reflect either a fault or lightning. In case of line switching, we don’t divide the line but insert switches at the both ends of the. This is the first step in BatchProcessATP function. It makes generic changes to ATP file. These generic changes will be explained in the next subsections shortly. After these generic changes are made to the ATP file, the file is saved in the toolbox directory with a temporary name.

The function BatchProcessATP then enters a for loop to create the cases the user wants to create. Having enetered the for loop, the fucntion consults the structured returned by the function CreateScenarios to make specific changes to a copy of the temporary file hitherto saved in the toolbox directory.

Having made the ATP file ready for the new file, the BatchProcessATP calls the runATP.exe from within Matlab to create an EMTP case. After than the utility PL42mat.exe is also called from Matlab to convert PL4 files to mat files. The function BatchProcessATP keeps executing the for loop till it reaches the last case after which is displays a message that all cases are created succissfully is no errors occur during simulations. The last step is deleting any temporatry ATP files files that were created during simulations. The over all logic of BatchProcessATP is shown in fig. 2. In the coming subsections, we explain the inner working of the functions CreateFault, LightStrike, and SwitchLine.

## Creation of fault cases (CreateFault function)

Although the function CreateFault is an internal function called by BatchProcessATP in its for loop, the user can use the function to create a single fault case. In this section we describe the logic implemented in this function.

As pointed out above the first case in creating a fault on a line is to divide the line into two sections and then inserting a generic fault block. The fault block inserted is given in fig. 3. The fault block simply consists of a phase splitter, two switches and resistances. The phase splitter is important to carry out different fault types. The block has two switches one connecting the common point of the two line sections– called by default F1- to the phase splitter and other provides connection to ground. The fault block is initially inserted with switch times and resistance values such that it is an open circuit to the simulation. When the user inputs certain type of fault, the function CreateFault changes those values. Once the block fault is customized to the fault type requested by the user the function CreateFault proceeds to next step.

The next step is actually dividing the line under study into two sections. For this, we created a search function that goes in the ATP file and find the specific line to be studied and then duplicate that line. The midpoint is always called F1. An example will make this clearer. If we are running fault cases on line connecting buses 11 and 17, then after the above step the line after being divided into two sections and the fault block inserted will look like what is shown in fig. 4.

After the generic block has been inserted and customized, the line divided into two sections, what is remaining to insert probes at both sides. We have another function that inserts such current and voltage probes at both ends of the line. One has to note that duplicating lines and adding probes will cause runtime errors if we didn’t change the names of the nodes of both lines. For this, we have another function that changes the names of the nodes once the line is duplicated. Completing the above example, after the probes are inserted, we have what is shown in fig. 5.

The last part of this function is adjusting the lengths of both sections. Another function is written for this purpose. The function takes its input from the data structure created by CreateScenarios function.

Having changed both line sections lengths, the modification of ATP file is now complete. The new file is saved under a temporary name in the toolbox directory. After we exit this function, the toolbox calls ATP from within Matlab, waits for simulations to be done, call PL42mat then deletes this temporary file. Giving the call back to the original for loop in the BatchProcessATP function, the function starts over till all cases are simulated. In summary, the function inserts a generic fault block, adjusts its parameters to correspond to the fault case, splits the line into two sections, renames the nodes of those sections, adjusts the length of the same two sections, calls runATP.exe and finally calls Pl42mat.exe to convert the pl4 output file of ATP to mat files to be used with Matlab.

## Creation of lightning case (LightStrike function)

Once again the LightStrike function can be used outside the BatchProcessATP function to create one lightning case. We now proceed to describe the logic implemented in this function.

As has been stated above, creating a lightning case involved dividing the line being hit into two sections and inserting a generic lightning block. The generic block being inserted is shown in fig. 6. The block when inserted function as open circuit by opening the switches F1-FA, F1-FB, F1-FC. The block as shown consists of three switches and three Heidler type lightning sources. Once the block is inserted, the function reads the data structure created by CreateScenarios function. The switches opening and closing times, the rising and the tailing times, and the amplitude of the strike are all adjusted according to the data structure being created by CreateScenarios function.

Following the same lines of thought as we did with creating faults, we divide the lines into two sections. After divinizing the lines into two sections, we insert probes at both ends, rename all the nodes and at the end modify the length of both sections to complete modifying the ATP file. As an example, if we want to create a lightning strike on line 11-17 then fig 7 shows the last form of the line after we inserted the lightning block, probes and renamed all nodes.

## Creating Line Switching cases (SwitchLine function)

Creation of line switching cases is the easiest and the most straightforward task. What is needed is not insertion of generic fault and lightning blocks nor dividing the line into sections but only insertion of two switches at the both terminals, current and voltage probes at both ends.

First, a special function inserts both switches at both ends of the line. After that the call is passed to another function that inserts current and voltage probes at both ends. After that both switches’ opening and closing times are adjusted according the data structure created by CreateScenarios function. At last, nodes are being renames not to cause error when running runATP.exe while ATP is building the connectivity matrix. An example is when we try to energize line 11- 17. The block corresponding to the above operation is given in fig. .

# Toolbox Flexibility and output file names

In this section we highlight some features of the toolbox that emphasizes its flexibility. We also show how the toolbox gives names to its output mat files.

Up till now very little has been said regarding how generic fault and lightning models are inserted. The lightning and fault generic blocks are actually copied from a text file shipped with the toolbox to the temporary ATP file creating in the course of execution of CreateFault and LightStrike functions. This gives the user the flexibility in inserting whatever he wants in the ATP file. The flexibility comes from the fact that those generic blocks are not hard coded in the toolbox. If the user for example wants instead of using a Heidler type lightning source to use another source, he can modify the text file corresponding to the generic lightning block accordingly. However, the user still needs to modify the function changes the parameters of the lightning strike.

Toolbox flexibility stems from the search functions it contains. The toolbox has built in search functions that can locate certain nodes, lines or sections in the ATP file. The search functions are used to locate any component but the user has to supply the names of the two terminal of the device. The functions can locate lines, switches and probes based only the names of the nodes of those components.

A systematic way to rename the output mat files is also offered. For faults, any mat file is of this form “Case\_LineUnderStudy\_FaultType\_FaultResistance\_FaultIncipientAngle\_FaultDistance”. For example an AG fault on line 11-17 at a distance of 60% from node 11 and an incipient angle of 30degrees and fault resistance of 10 Ohms will have the following name: Case\_11-17\_AG\_10Ohms\_30d\_60percent .

For lightning on the other hand the following naming conversion is offered: Case\_ LineUnderStudy\_PhaseBeingStroke\_LightningAmplitude\_IncipientAngle\_Distance. For example a lightning staking phase A of line 11-17 having an amplitude of 5000Amperes at a distance of 60% of length from bus 11 at a 30 degrees incipient will have the following name: Case\_11-17\_5000A\_30d\_60percent.

For line switching the following naming conversion is being offered: Case\_LineUnderStudy\_Angle\_Distance, where distance is either 0 or 100 based on which end being switched. For example if we energize line 11-17 using the end 11 then distance is zero else it is 100.

##### Acknowledgment

The author would like to acknowledge the help of Po-Chen Chen of Texas A&M University for his help solving many ATP related problems.

##### References

1. M. Kezunović, Igor Rikalo, D.J. Šobajić, High-speed fault detection and classification with neural nets, *Electric Power Systems Research*, Volume 34, Issue 2, August 1995, Pages 109-116
2. Gaouda, A.M.; Kanoun, S.H.; Salama, M.M.A.; Chikhani, A.Y., "Wavelet-based signal processing for disturbance classification and measurement," *Generation, Transmission and Distribution, IEE Proceedings-* , vol.149, no.3, pp.310, 318, May 2002
3. Bo, Z.Q.; Jiang, F.; Chen, Z.; Dong, X.Z.; Weller, G.; Redfern, M.A., "Transient based protection for power transmission systems," Power Engineering Society Winter Meeting, 2000. IEEE , vol.3, no., pp.1832,1837 vol.3, 23-27 Jan 2000
4. Martin, F.; Aguado, J.A., "Wavelet-based ANN approach for transmission line protection," *Power Delivery, IEEE Transactions on* , vol.18, no.4, pp.1572,1574, Oct. 2003
5. Ibrahim, W.R.A.; Morcos, M.M., "Artificial intelligence and advanced mathematical tools for power quality applications: a survey," Power Delivery, IEEE Transactions on , vol.17, no.2, pp.668,673, Apr 2002
6. G. D. Guidi-Venerdini; F. E. Pérez-Yauli, “MATLAB Program for Systematic Simulation over a Transmission Line in Alternative Transients Program, ” International Conference on Power System Transients IPST 2013, Vancouver, BC, Canada, July 18-20, 2013
7. Bonneville Power Administration, "Alternative Transients Program

(ATP)," ed. Portland, Oregon, U.S.

1. [online] https://bitbucket.org/ahmadmabdullah/atpmat
2. M. Ceraolo and R. Salutari, "PL42mat," ed. Pisa, 2009.
3. A Abdullah, A Esmaeilian, G Gurrala, P Dutta, T Popovic, M Kezunovic, “Test Bed for Cascading Failure Scenarios Evaluation”, International Conference on Power System Transients IPST 2013, Vancouver, BC, Canada, July 18-20, 2013
4. Power Systems Test Case Archive, [online] <https://www.ee.washington.edu/research/pstca/pf118/pg_tca118bus.htm>
5. http://www.emtp.org/