ATPMat®: An Open Source Toolbox for Interfacing ATP and Matlab®

Ahmad Abdullah

***Abstract*--ATPMat® 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. The toolbox provides a way not only to change one ATP file but to batch process ATP files to create a batch of faults, line energisation cases or lightning cases for a certain 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. Both Windows and Linux implementations are presented along with the setup of environment variables necessary to run ATP in batch mode.**

***Keywords*: Index Terms--- ATP, EMTP, Data Translator, MATLAB, Open Source Toolbox, Faults, Lightning, Line Switching.**

I. 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. In [1] 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 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 [2] is used to automate creation of fault cases, line energisation 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 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. 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 the 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 to be used in Matlab®. Windows and Linux implementations are offered as part of the toolbox. The user needs to adjust the settings of ATP before using the toolbox. The code has been tested on an Intel core i7 processor and on the Centos 7. The paper builds upon a paper that we published before in IPST 2011 [3]. 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 models for IEEE 118 case as a test system. The paper is organized as follows: Section II discusses how to set up system environment for both Windows and Linux. Section III explains about the structure of the toolbox and the functions supported by it. Section IV contains the conclusion and future work.

Ahmad Abdullah is a PhD Candidate at Electrical and Computer Engineering at Texas A&M University, College Station, TX, 77840 (e-mail: ahmad@tamu.edu).

Paper uploaded to Bitbucket/ahmadmabdullah/atpmat. This paper is under review in a conference.

II. Setting up system 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 [4]. 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 [2] and then click on make. This completes the setting up of environment in Windows. In Linux, the user still has to get the Linux implementation of the aforementioned files but them in one folder then adjusts the Listsize and save the directory for further use.

III. Toolbox Structure

The overall toolbox structure is shown the flow chart in fig. 1.The main function of the toolbox is BatchProcessATP. This is the function that's responsible for creating a batch of ATP simulations. 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 a structure that contains the parameters the simulation.

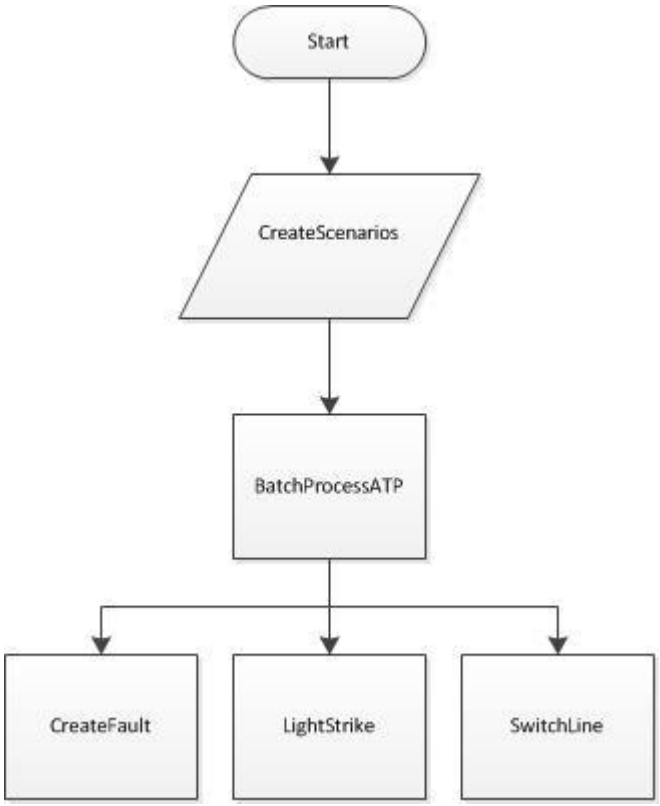

Fig. 1. Overall toolbox structure

The type of the study has to be one of three options: “fault”, “lighting” or “energisation”. The structure that is supplied to BatchProcessATP is created using another function called CreateScenarios. The information in the structure will vary according the study sought. If the user wishes to create a batch of faults, she needs to supply the following: fault distance, fault resistance, fault type, line under consideration, 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 below.

A. Creation of fault cases (CreateFault function)

After the user specifies the parameters of the simulation in the CreateScenarios function, the function BatchProcessATP calls the function CreateFault repeatedly. For each call, the function CreateFault reads the original ATP file, copies it to another temporary file with a coded name and inserts the block in fig. 2 in the newly created ATP file to create a fault. The switch F1-F1S in the block is then controlled to be on or off depending on the time the user enters. The switch F1N-GND is switched on depending whether the user wishes to create a ground fault or not. The line that is supported in distributed line and in the future LCC will be supported. The

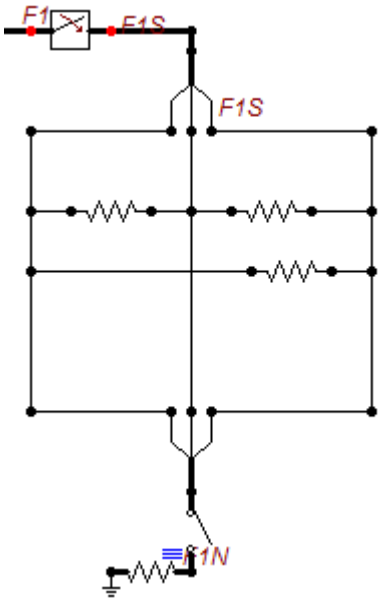

Fig. 2. Fault Block inserted in ATP file for fault creation

block is created such that adjusting the value of the resistors will create different types of faults. The resistor are adjusted such that when the resistance is 1e+6, an open circuit is

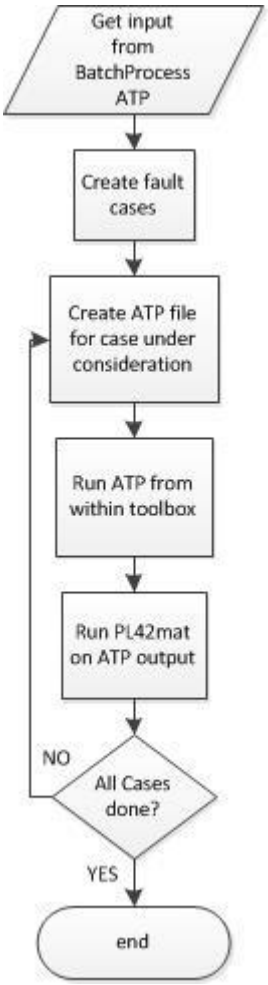

Fig. 3. Creation of fault cases

created between the phases connecting the resistor, while when the value of the resistor is 1e-6 a short circuit is created between the phases that connects that resistor. The toolbox reads the length of the line from the ATP file and then splits the line into two parts the sum of the lengths of which is equal to original line. The mid-point is renamed by default to F1. The lengths of both sections are then adjusted to create the case specified by the user. Voltage and current measuring probes are also inserted at the terminals of the line. ATP is then called from within Matlab to create a Pl4 file which is converted to mat file by the virtue of Pl42mat utility. The overall flow chart is shown in fig. 3.

B. Creation of lightning cases (LightStrike function)

The logic behind automatic creation of lightning cases follows closely the one used for creation of fault cases. The function BatchProcessATP calls the LightStrike function repeatedly. For each call, a temporary copy of the original ATP file is created. A block that is shown in fig. 4 is inserted in the temporary ATP file. The toolbox again reads the length of the line to be stroked by lightning and divides the lines into two halves, the midpoint of which is renamed by default to F1. The switches F1-FA, F1-FB and F1-FC in the shown block are modified according to the needs of the user to strike one or

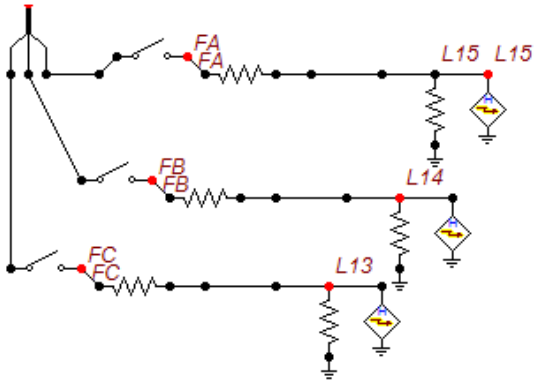

Fig. 4. A block to be inserted in ATP to create a lightning strike

more phases by a lightning a strike. A Heidler component shunted by a suitable resistor and in series with another appropriately sized resistor is used to simulate lighting. The user can adjust the amplitude and parameters of that lightning waveform according to her need. After the ATP file has been prepared, ATP is called to produce a PL4 file and Pl42mat is then called to convert that file to mat file suitable to use in Matlab. The overall flowchart summarizes what the function LightStrike when used in context with creating lightning cases is shown in fig. 5.

C. Creation of line energisation cases (SwitchLine)

The procedure here again resembles what has been done with the above two cases. BatchProcessATP calls SwitchLine many times. For each call, a temporary ATP file is created according to the user input. Two switches are inserted; one at each terminal of the line. The side to be switched is selected for switching. ATP is then called from within Matlab to produce a Pl4 file which is converted to Pl42mat using

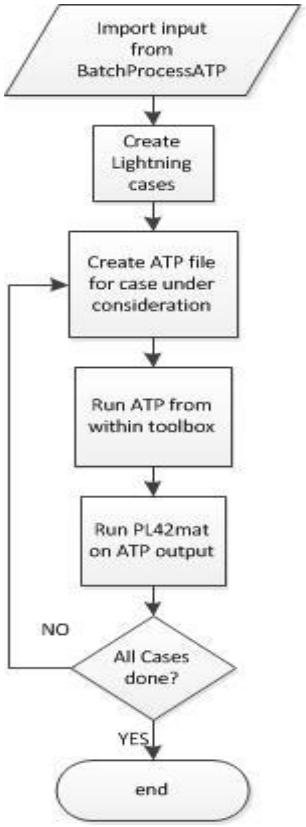

Fig. 5. Automatic creation of lightning cases in ATPMAT

Pl42mat utility. The overall methodology is the same as the one given before for generation faults and lightning strike cases so we choose to omit the flow chart for line switching.

IV. Conclusions and Future Work

We presented in this paper some useful functions and pieces of code that can be used by researches and engineers to automate many tasks. The main contribution of the paper is the following:

- Creation of batch of fault cases.
- Creation of batch of lighting cases.
- Creation of batch of line energisation cases.

Future work will include more support for more types of transmission line and more types of lightning sources. More flexibility in renaming the fault node will be added in future versions.

V. 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.

VI. References

[1] G. D. Guidi-Venerdini; F. E. Pérez-Yauli, “MATLAB Program for Systematic Simulation over a Transmission Line in Alternative Transients Program,” IPST 2013, Vancouver, BC, Canada, July 18-20, 2013.
[2] https://bitbucket.org/ahmadmabdullah/atpmat.
[3] A Abdullah, A Esmaeilian, G Gurrala, P Dutta, T Popovic, M Kezunovic, “Test Bed for Cascading Failure Scenarios Evaluation”, IPST 2013, Vancouver, BC, Canada, July 18-20, 2013
[4] http://www.emtp.org/