N-1Analysis Explanation

7/28/2022

This code provides an example of how SimAuto can be used to automate contingency analysis. It is intended for training purposes, but it can potentially be adapted to fit your project's needs. If you update this code in a way that others may find helpful, feel free to send your work to support@powerworld.com and we may post it on our site.

In N – 1 Contingency Analysis, PowerWorld analyzes how the grid reacts when any single element is taken offline. This code performs N – 1 analysis under increasing load – analyzing what happens to the case when any branch is removed, then increasing the case's load by a given percent and repeating. All contingency data from each load level are loaded into CSV files, which are later consolidated into files for each branch that exceeds its limit during the analysis. These consolidated CSV files allow users to create charts showing how the branch responds to a given contingency under increasing load.

Information on installing SimAuto can be found at Introduction to PowerWorld Simulator: Interface and Common Tools, beginning on the 18th slide. Some example SimAuto commands can be found at Example SimAuto Files » Knowledge Base » PowerWorld (and also throughout the rest of the slideshow containing setup information). Finally, for a library of script commands (which contribute immensely to SimAuto's usefulness), see General Format of Auxiliary Data Files (powerworld.com). While looking through this code, it can be useful to have your case simultaneously open in PowerWorld Simulator, where you can view data and run the script commands found in this program.

N-1Analysis.py begins by importing several tools, most notably SimAuto. After importing win32com.client, SimAuto is assigned to the variable 'simauto_obj' – so all SimAuto commands can be called with simauto_obj.FUNCTION. The code then requires several variables to be set by the user: the path to the case you wish to analyze, the path to that case's folder, the maximum desired load scale factor, and the amount you want the load scale factor to increase each iteration.

The program then begins the first stage of its process: the N – 1 contingency analysis. The first step in this process is defining a scale() and analyze() function, which the program will use in this first stage. Each bus in the case has a parameter, SCALE, that designates if the bus should be scaled with the rest of the case. The scale() function begins by setting all bus's SCALE parameters to YES, but additional filters can be added to only scale part of the case. Scale() then scales the case by the factor that was passed as the argument to this function. Finally, this new scale is saved as the reference case – otherwise, the scale would be undone before contingency analysis is performed. Notice how reliant this process is on simauto_obj.RunScriptCommand('COMMAND): see General Format of Auxiliary Data Files (powerworld.com) for more information on how to use each of these script commands.

The next function, analyze(), begins by auto-inserting a contingency for each branch in the case (meaning that it will check what happens when each branch goes offline one at a time). Note that this is a two-step process: before running the script command CTGAutoInsert;, the object Ctg_AutoInsert_Options must be set to ensure that the correct contingencies are being inserted. Analyze() then solves each contingency and saves the data to a CSV file.

With these two functions defined, the remainder of the N-1 Contingency Analysis becomes simple. Beginning with a scale factor of 1 and increasing to ‘max’, the function scales the case, performs contingency analysis and saves to a CSV file, undoes the scaling, and repeats.

The second stage of this process, in which the function combs through the CSV files and consolidates them into one file for each limit-exceeding branch, uses no SimAuto or script commands – as such, the functions used here can be best explained in-depth by documentation for python’s CSV library. This stage begins with a loop that iterates through each file created by the N – 1 analysis performed earlier. The contents of each CSV file are read into violation_data, a list containing a 2-dimensional matrix with all information from the CSV files. If the case is N – 1 secure at a certain load, the file will be empty – if it is not, the function goes through each column to find each branch that violated its limit during the analysis (because each column in the initial CSV files represents a violating branch). If it has never seen that particular branch before, the program writes headers for each column of a new file. The code then takes the data from that column (the load percentage, limit percentage, and contingency name) and reads them into a new file. It then advances to a new column, then a new row, and ultimately a new contingency file. This process yields CSV files that, when opened in excel, can create helpful charts that show how each branch reacts to certain contingencies at different loads.

To create charts, find the consolidated CSV files within the folder containing the case you analyzed (they will have a name like ‘One (1) to Three (3) CKT 1’). Open one in excel, highlight all the data, then click Insert → Pivot Chart and click OK on the dialog box that opens. On the right of your screen, you see each field – drag ‘Load Scale Percentage’ to ‘Axis (Categories)’, ‘Percentage of Limit’ to ‘Values’, and ‘Contingency Name’ to ‘Filters’. Close the editor. If you click in the upper right corner of your chart, you can select different contingency names, thus allowing you to view how the branch you’re analyzing responds to a given contingency under increasing loads.