Transient Stability Analysis with PowerWorld Simulator

T13: Transient Stability Play-In Signals and Script Commands

PowerWorld Corporation

2001 South First Street
Champaign, Illinois 61820
+1 (217) 384.6330

support@powerworld.com
http://www.powerworld.com

Play-In Signals

- Simulator supports user-configurable time-series blocks as inputs to the transient stability simulation
- These models are called **Play-In Signals**
- **Play-In Signals** encapsulate custom time series fields which can be analyzed and plotted like any other signal in Simulator
- Any number of play-in signals can be defined

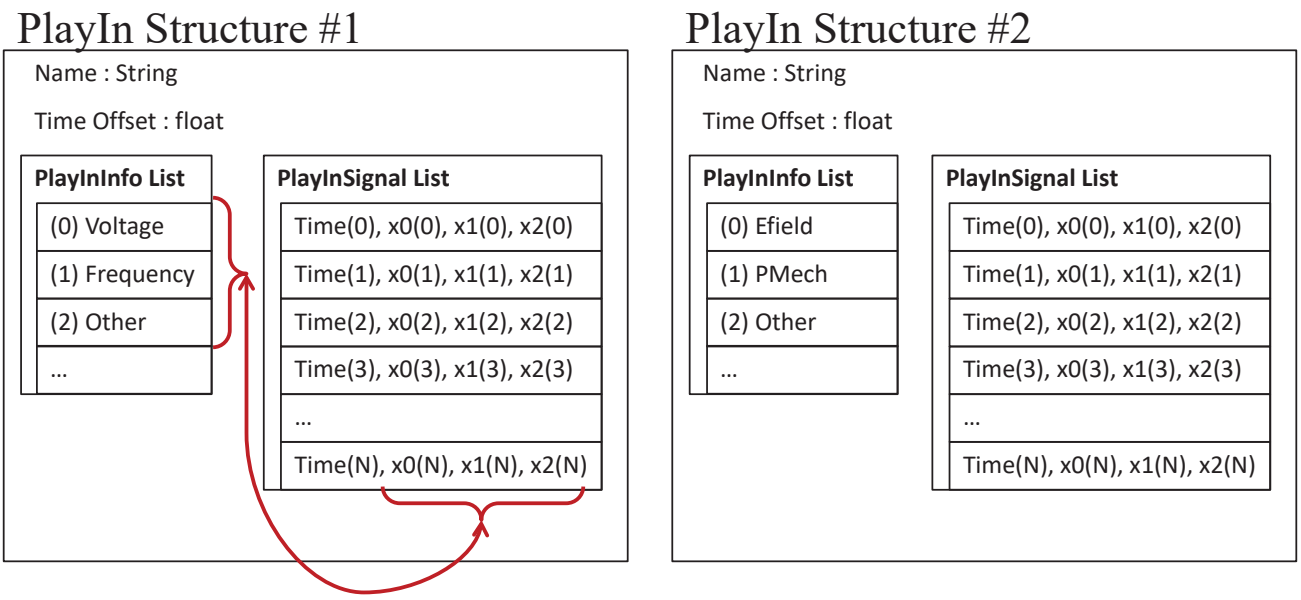
Two Purposes of Play-In Signals

- Define a signal for plotting purposes
  - Just another signal to put on a plot for comparison purposes only
  - A way to load and view data independently
- Play-in of model data
  - Add "*Machine*" model: play-in a bus voltage and frequency
  - Add "*Exciter*" model: play-in field voltage
  - Add "*Governor*" model: play-in mechanical power
  - Add an auxiliary other machine model for governor reference (P_{ref}) or exciter reference (V_{ref})

Play-In Model Structure

- Three new data objects
  - **PlayIn** – a named structure that contains the other two objects
    - *Name, Time offset*
  - **PlayinInfo** – a list of information about the signals contained in one PlayIn structure
    - *Name, Scale, Offset, Filter Time, Signal Index*
  - **PlayinSignal** – a time-series list of numerical data for one PlayIn structure
    - *Time, List of values for this time*

Graphical Representation of Play-In Structures



3 PlayIn Info objects → 3 values for each PlayInSignal

Fitting a Play-In Signal to Simulation

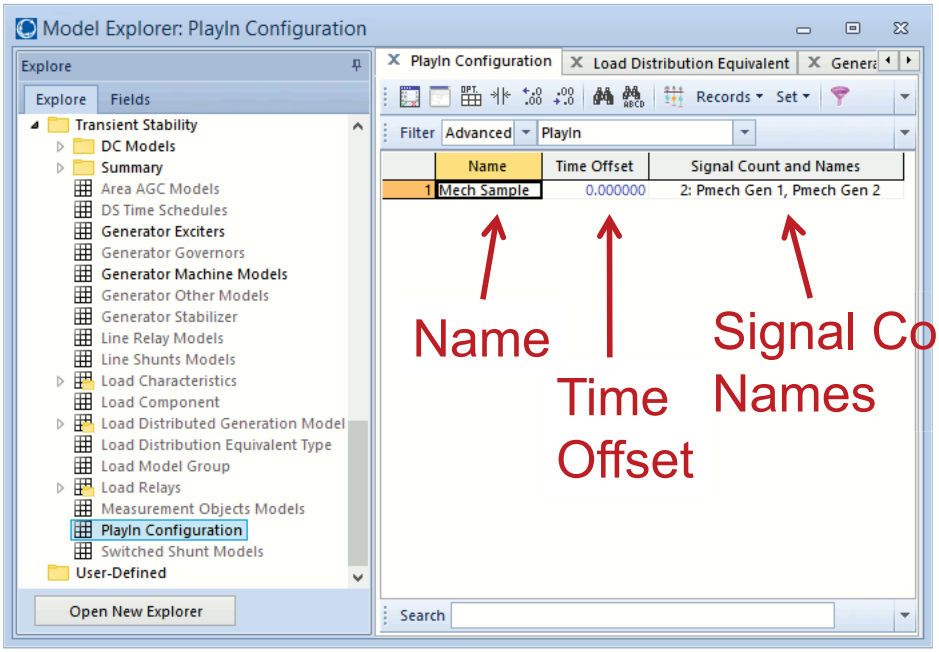
- **PlayIn** structure
  - *Time Offset* : shifts signal in time axis to match simulation
- **PlayInInfo**
  - *Offset* : shifts the signal in y-axis
  - *Scale* : multiplies the signal
  - *Filter Time* : runs the signal through an additional [1/(1+Ts)] delay block during the simulation
- General note about time
  - All signals in a **PlayIn** structure use the same time axis
  - For all signals, a value must be specified at every time
  - Use multiple PlayIn structures if your signals do not have the same time points

Play-In Example

- Open TS9Bus Bus Fault PlayIn Setup.pwb
- This case contains a PlayIn object which is not yet linked to anything in the Simulation
- Open the PlayIn Configuration tab on the Model Explorer

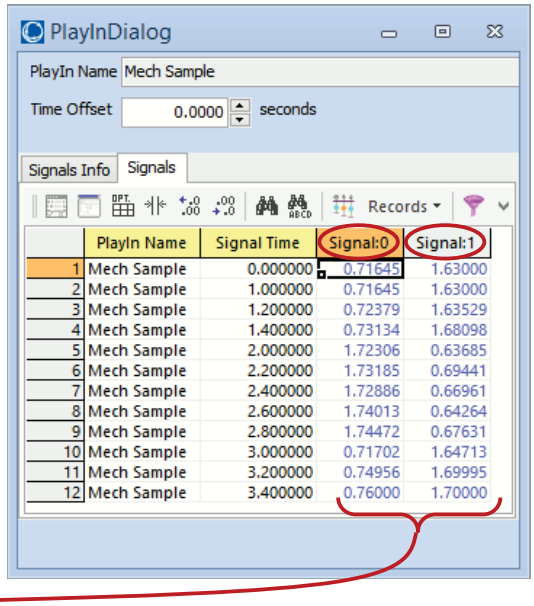
Play-In Configuration

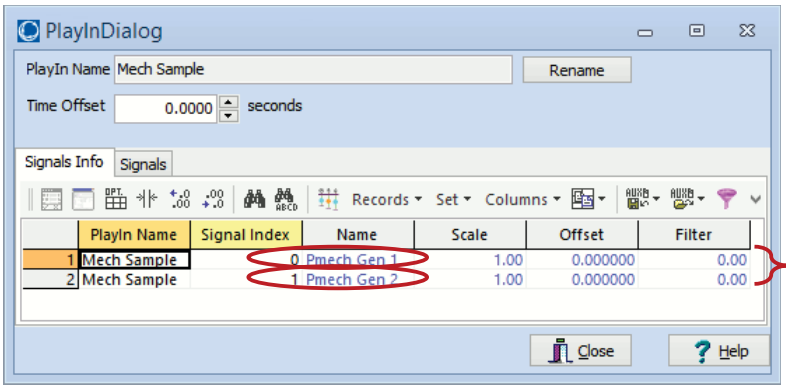
- Use this tab to specify new PlayIn objects



Pieces of a Play-In: Signal Info, Signals

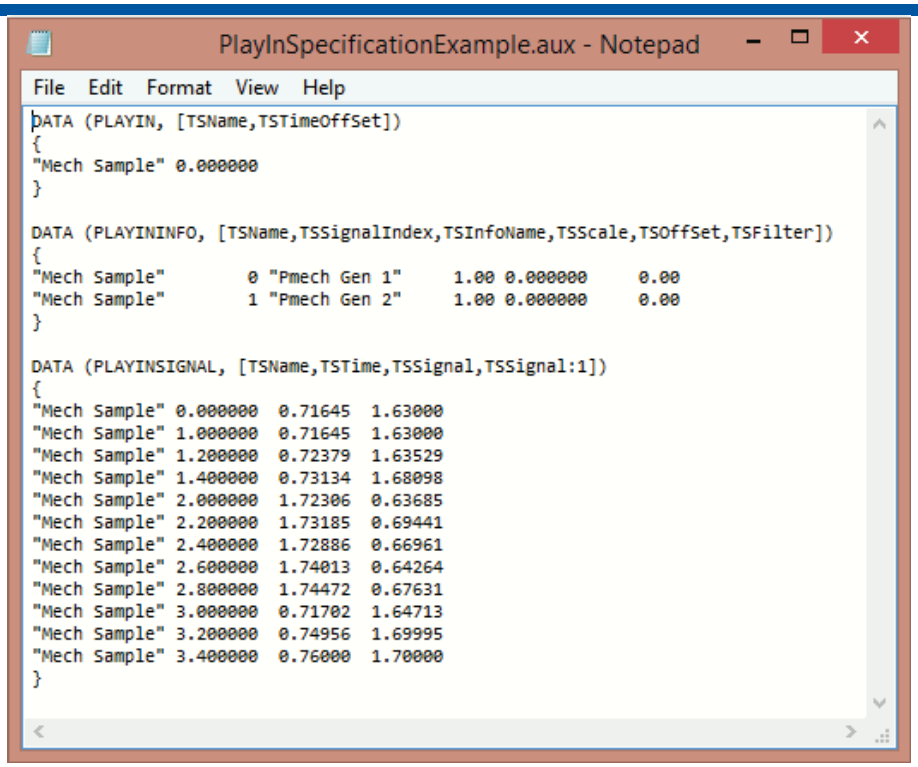
- Each PlayIn object contains
  - Signal Info specifications
  - The signals themselves





This PlayIn contains two signals listed by signal index

Play-In Specification as an AUX File

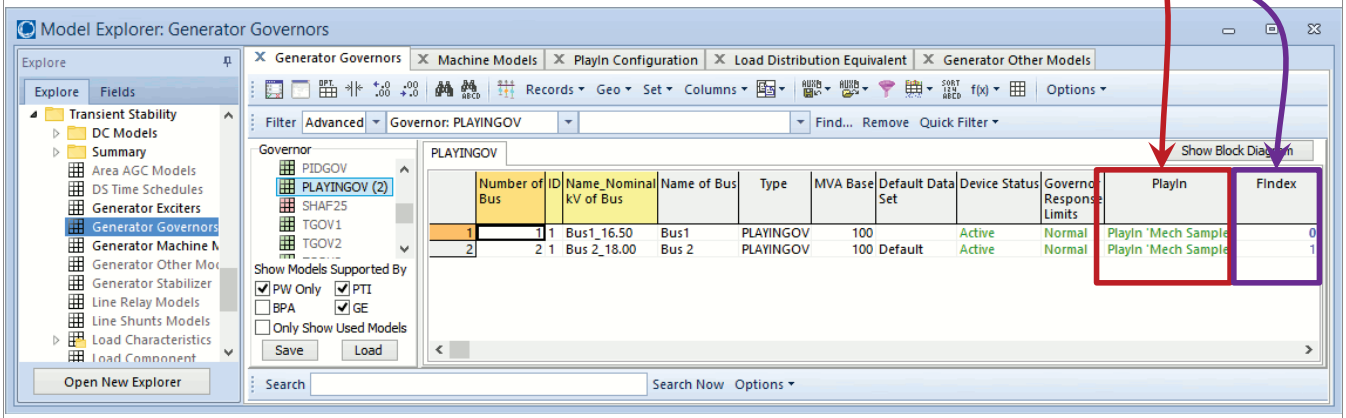


New Dynamic Models

- **PLAYINGEN**: “Machine” model
  - play-in a bus voltage and frequency
- **PLAYINEX** : “Exciter” model
  - play-in field voltage
- **PLAYINGOV**: “Governor” model
  - play-in mechanical power
- **PLAYINREF**: “Other” generator model
  - play-in governor reference (*Pref*)
  - play-in exciter reference (*Vref*)

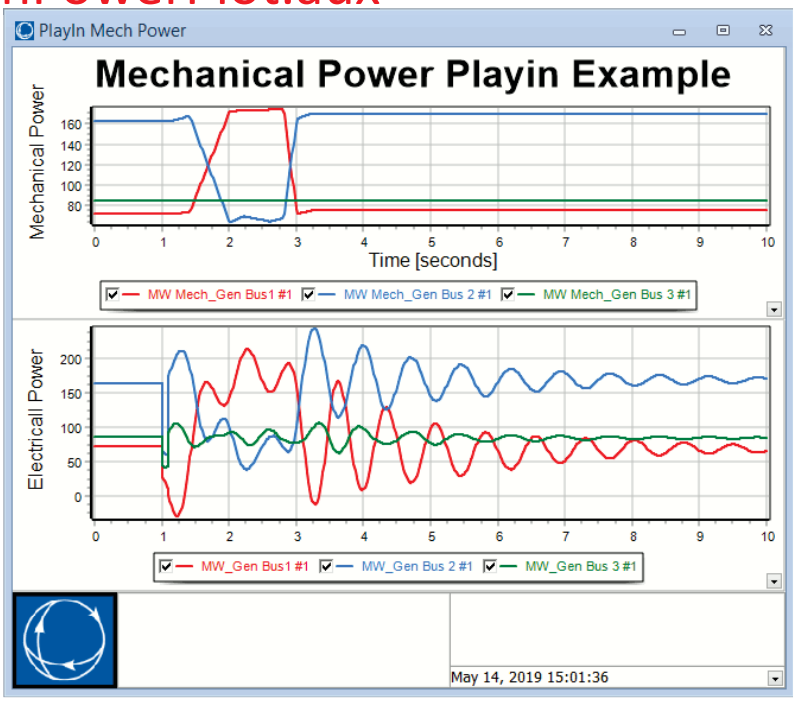
Example: GOVPLAYIN

- Add a new governor model “PLAYINGOV”
  - Specify which PlayIn object to use
  - Specify the signal index to refer to



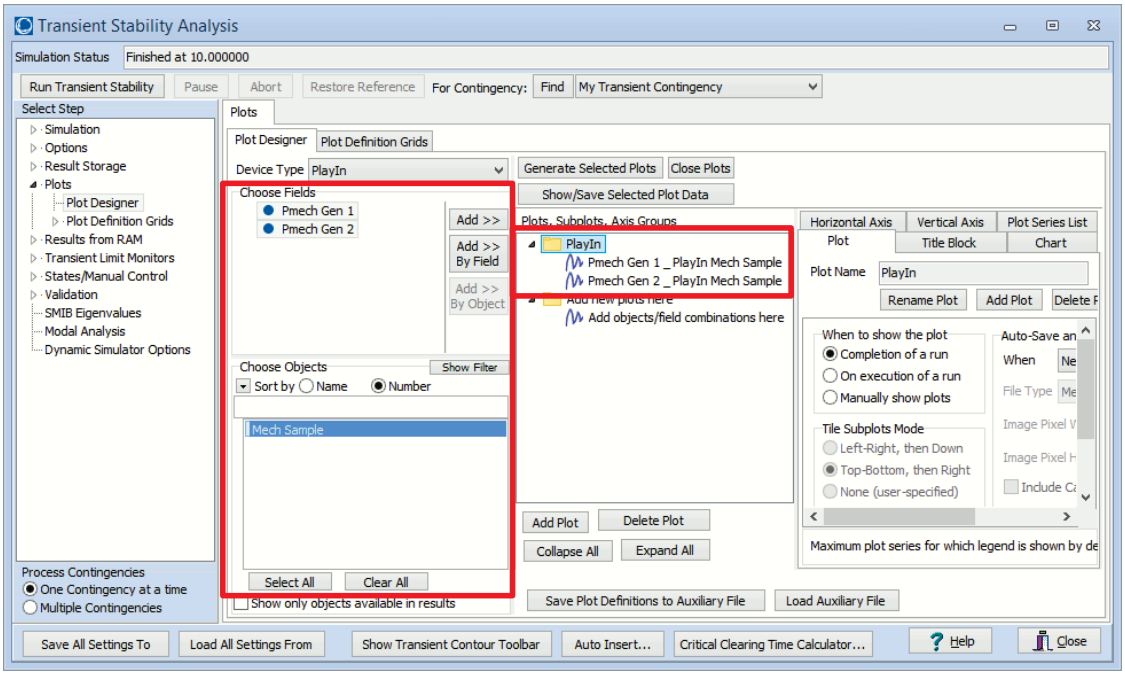
Example Simulation

- Load playInMechPowerPlot.aux
- Top curves show the played in mechanical power



Play-In for Plotting

- Directly plot a play-in signal on a plot



Transient Script Commands

- Script commands provide flexibility for calling the functionality of Simulator remotely
- Much of the transient stability functionality is supported by script commands
- Combining the powers of aux files, script commands, and SimAuto, the possibilities for managing and controlling your simulation expand greatly
  - Here we assume some basic familiarity with pre-existing Simulator concepts including SimAuto, aux files, and script commands
  - The focus in this section is on how to ***use*** the transient stability script commands to facilitate running your simulations and obtaining the results

Available TS Script Commands

- **TSSolveAll;**
  Solve all of the transient contingencies.
  Example usage: TSSolveAll;

- **TSSolve("ContingencyName", [StartTime, StopTime, StepSize, StepInCycles]);**
  Solve a particular transient contingency. The specification of values for StartTime, StopTime, StepSize, and StepInCycles is optional. If these are not specified, the default values for the specified contingency are used. The StartTime and StopTime of the simulation are specified in seconds.
  Example usage(s): TSSolve("My Transient Contingency");
  TSSolve(My Transient Contingency);
  TSSolve(ctg2, [0.0, 10, 0.5, Yes]);
  TSSolve(ctg2, [1.0, 9, 0.5, Yes]);
  TSSolve("ctg2", [1.0, 9, 0.01, No]);

- **TSWriteOptions("FileName",[SaveDynamicModel, SaveStabilityOptions, StabilityEvents, SaveResults, SavePlotDefinitions,KeyFieldStatus]);**
  Write the transient stability options to a specified auxiliary file. The parameter values are optional. Each may be set to Yes/No/"". If the bracketed parameter is omitted, all of the options will be written out.

Available TS Script Commands

- **TSLoadGE("FileName", Yes/No);**

  Load GE dynamic data. The FileName is the name of the GE dynamic file (*.epc). The option indicates what to do if a GENCC record is found, where “yes” automatically splits the existing generator into two generators to accommodate this type of model.

- **TSLoadPTI("FileName", "MCREfilename", "MTRLOfilename", "GNETfilename", "BASEGENfilename");**

  Load PTI dynamic data. "FileName" is the name of the PTI dynamic file (*.raw), "MCREfilename" specifies a file for splitting generators based on an MCRE *.RWM data file, "MTRLOfilename" specifies a file for splitting out motor loads based on an MTR_LD *.dat data file, "GNETfilename" specifies a file for making generator dynamic models inactive based on a GNET *.idv data file, and "BASEGENfilename" specifies a file for setting the Governor Response Limits based on a BASEGEN *.dat data file.

- **TSLoadBPA("FileName");**

  Load BPA dynamic data. "FileName" is the name of the BPA dynamic file (*.swi).

Available TS Script Commands

- **TSCalculateSMIBEigenValues;**

  Calculate the single machine infinite bus eigenvalues.

  Example usage: TSCalculateSMIBEigenValues;

  Save out eigenvalue results using the SaveData script function after the eigenvalues have been calculated.

  Example usage: SaveData(“C:\Support_SimAuto\eigsOut.aux", AUXDEF, PWTXGEN, [BusNum, BusName, GenID, TSEigenValueValid, TSNumEigenValue, TSNumEigenValueZero, TSEigenValueMax, TSEigenValueMin, TSComplexEigenValue, TSComplexEigenValue:1, TSComplexEigenValue:2, TSComplexEigenValue:3, TSComplexEigenValue:4, TSComplexEigenValue:5], []);

- **TSSaveTwoBusEquivalent("FileName.pwb",[Bus busNum]);**

  Use this to create and save out a two-bus equivalent, where the dynamic models at the specified bus are modeled in detail and the rest of the system is approximated as an infinite bus.

  Example usage: TSSaveTwoBusEquivalent("TwoBusTest.pwb",[Bus 2]);

TSGetResults Script Command

- **TSGetResults("FileName", Single/Separate, ["ctgname1", "ctgname2"], ["Plot 'plotname1'", "Plot 'plotname2'", "Bus busNum1 | busField1", "Bus busNum2 | busField2"], StartTime, StopTime);**

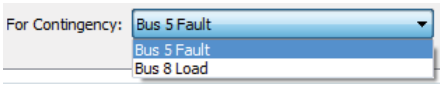
  Save out results for specific variables from plots, subplots, and object/field pairs after a transient stability simulation has been run. Single/Separate determines whether the results are all saved in one file with name "filename_Results.csv" or whether results for each transient contingency is saved in a separate file with name "filename_ctgname.csv." A separate header file is saved out, with a name of "filename_Header.csv". The StartTime and StopTime, in seconds, give the window of simulation time from which the results are to be retrieved. If not specified, results for the entire simulation time are obtained.

  Example usage(s):
  TSGetResults("filename.csv", SEPARATE, [B5Fault], ["Plot 'MW Output'", "Bus 4 | frequency"]);
  TSGetResults("filename.csv", SEPARATE, [ctg2, ctg3], ["Plot 'Gen_Rotor Angle'"], 0.0, 10.0);
  TSGetResults("filename.csv", SEPARATE, ["ctg3", ctg1], ["Plot 'Gen_Rotor Angle'"], 0.0, 10.0);
  TSGetResults("filename.csv", SINGLE, ["ctg2"], ["Plot 'Gen_Rotor Angle'"], 0.0, 10.0);

  Note: To save out the results, the results must EXIST! If there are no results you cannot save them out. Make sure you have set the "Results to Save" options or created plot definitions in Simulator and that you have run the simulation.

Review of Script Actions

- Open TS9BusMultipleContingency.pwb
  (or any case with contingencies inserted)



- Go to **Tools >> Script** or **Add Ons >> Script**

Example Scripts To Run

- **Try the following script commands**
- Solve both contingencies together or individually

```
TSSolveAll;
TSSolve("Bus 5 Fault");
TSSolve("Bus 8 Load");
```

The format of the input parameters matters and must be correct for your script to work correctly

It's a good idea to use a text editor to write and save the scripts for later use

- Save out the results

```
TSGetResults("resOut.csv", SINGLE, ["Bus 5 Fault"], ["Plot 'MW output'"], 0.0, 10.0);
```

- View the results

Using SimAuto with Transient Stability

- **TSGetContingencyResults(CtgName, ObjFieldList, StartTime, StopTime);**

The TSGetContingencyResults function is used to read transient stability results into an external program (i.e. Matlab or VB) using **SimAuto**, where they may be further processed.

Example:

TSGetContingencyResults("ctgname", {"Plot 'plotname1'", "Plot 'plotname2'", "Bus busNum1 | busField1", "Bus busNum2 | busField2"}, StartTime, StopTime);

Parameter Definitions

CtgName : String The contingency to obtain results from. Only one contingency at a time.

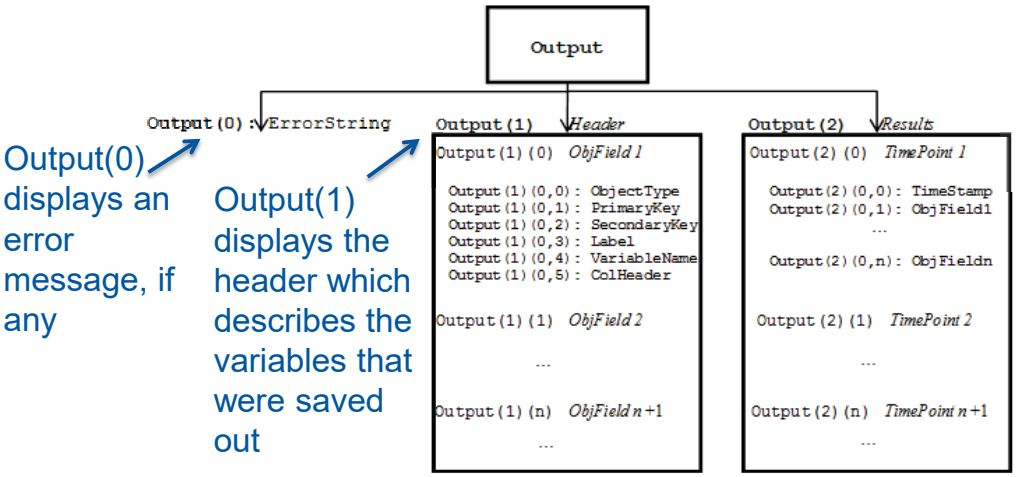
ObjFieldList : Variant A variant array of strings which may contain plots, subplots, or individual object/field pairs specifying the result variables to obtain.

StartTime: String The time in seconds in the simulation to begin retrieving results. If not specified, the start time of the simulation is used.

StopTime: String The time in seconds in the simulation to stop retrieving results. If not specified, the end time of the simulation is used.

SimAuto Transient Result Structure

The SimAuto output for this function is a Variant with a three-part structure.



Output(0) displays an error message, if any

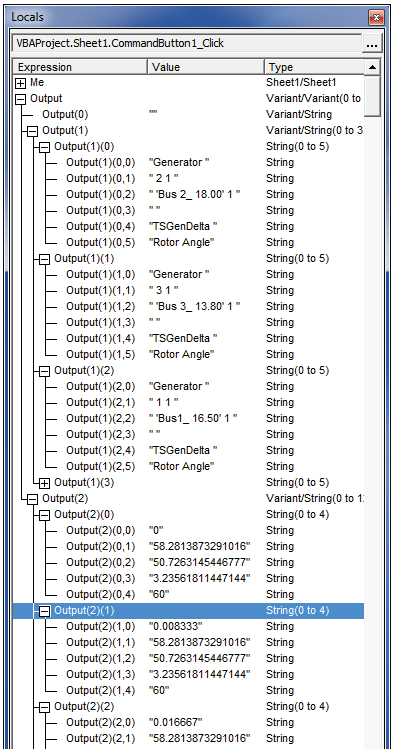
Output(1) displays the header which describes the variables that were saved out

Output(2) displays the time series numerical data for the variables in the header

TS Result Structure in SimAuto

SimAuto TS Results Example

- Open the "SimAutoTSTraining.xls" file
- If the Security Warning appears in the header, click **Options…** then choose "Enable Content"
- Make sure the Directory Path text (Cell B1) matches the proper location on your machine
- To view the code, go to **Developer >> Visual Basic**
- If the Developer tab is not present, you can add it
- Add some breakpoints; you'll see the results inside Visual Basic
- Click the run arrow or click on the "Run Multiple-Contingency Example" button



Visual Basic Example

SimAuto TS Results Example

'TSGetResults - Use a script command to save the results

```
Dim ScriptCommand2 As String
'ScriptCommand2 = "TSGetResults('scripttest.csv', SEPARATE, [Bus 5 Fault], ['Plot 'MW output'', 'Bus 4 | frequency']); "
Output = mySimAuto.RunScriptCommand(ScriptCommand2)
  If Output(0) <> "" Then
    'Error occurred
    DisplayErrorMessage Output(0)
  End If
```

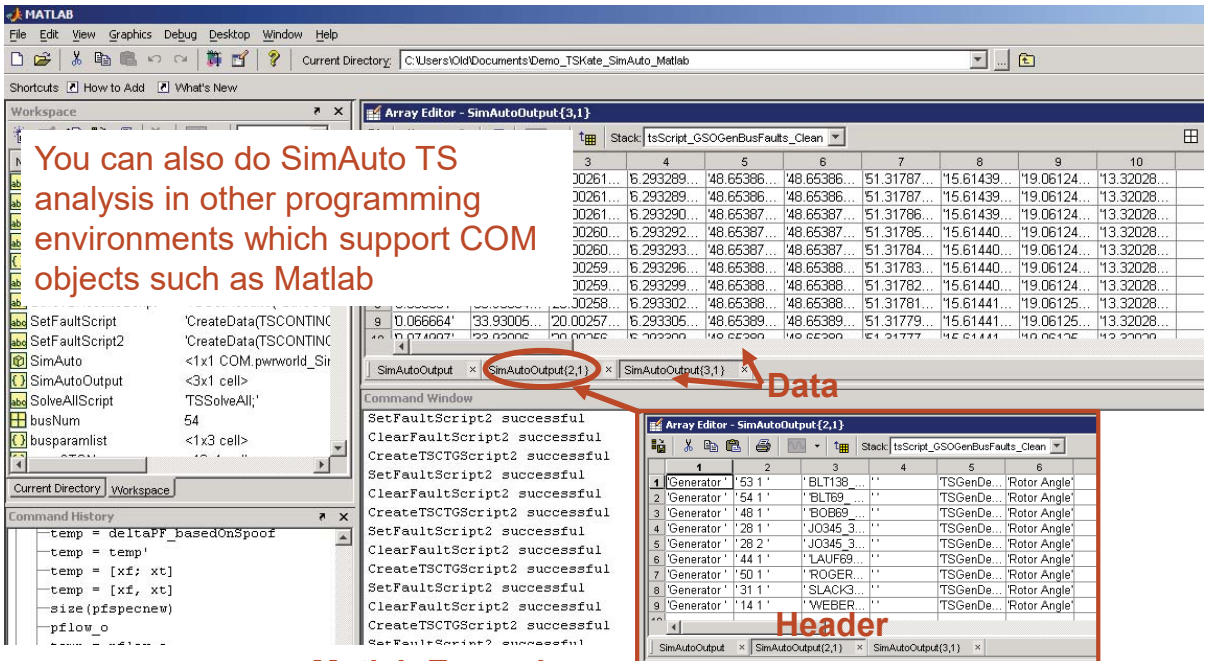
'TSGetContingencyResults – Directly retrieve the results into SimAuto

```
Dim objFieldList As Variant
objFieldList = Array("Plot 'MW output'", "Bus 4 | frequency")
Output = mySimAuto.TSGetContingencyResults("Bus 5 Fault", objFieldList, "0.0", "10.0")
If Output(0) <> "" Then
   'Error occurred
    DisplayErrorMessage Output(0)
  End If
```

Visual Basic Example

Play around with the code related to saving out the Transient Stability results

SimAuto TS Results Example



Matlab Example

Blank Page

Blank Page