Tutorial

PSCAD and MATLAB/Simulink Interface

Version 4.5

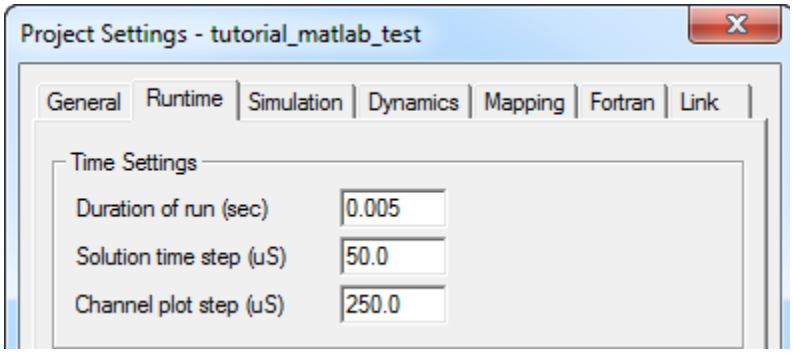
475 Wall Street, Princeton, NJ 08540, USA.

Index

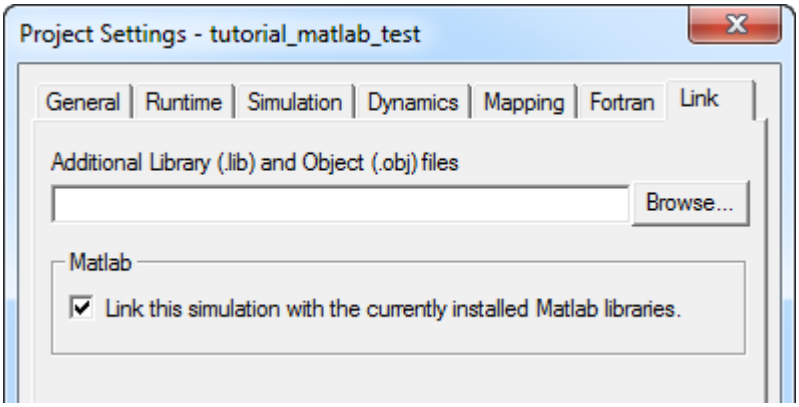
Tutorial 1

MATLAB block with addition and multiplication operations

1. After creating a new case in PSCAD, right click and select “Project Settings…” and adjust the “Duration of run (sec)” to be 0.005.



2. Select the “Link” tab. Check the box “Link this simulation with the currently installed Matlab libraries”. (This must be done for every new case that will contain a custom MATLAB component)

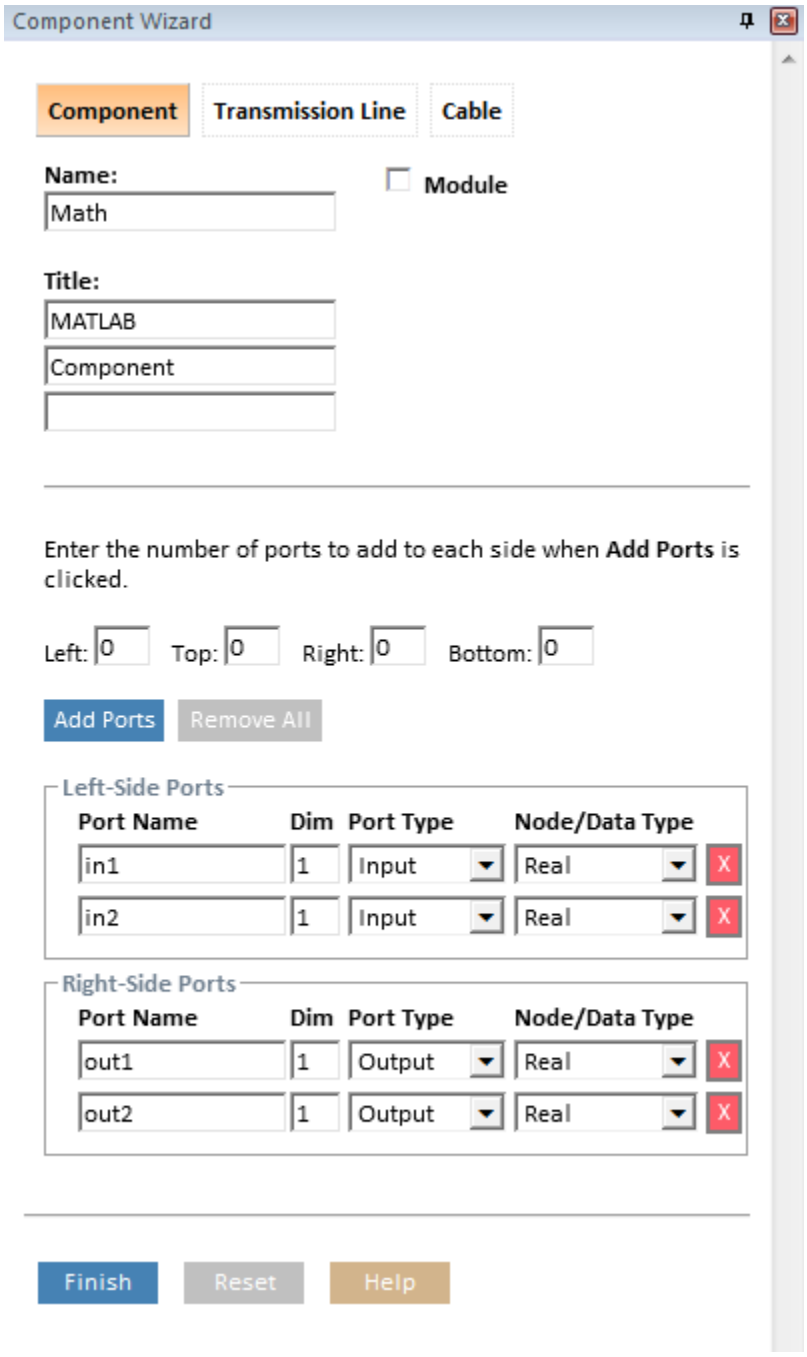


3. A new MATLAB m-file will be created to be integrated into a PSCAD schematic drawing. A simple example is shown below; the program has 2 input and 2 output variables. All of the inputs and outputs are floating point variables.

```
1    function [out1, out2] = add_mult(in1, in2)
2    % Simple MATLAB script to add and multiply
3    % together intefaced with PSCAD
4
5        % Add operation
6 -      out1 = in1 + in2;
7
8        % Multiply operation
9 -      out2 = in1 * in2;
```

4. Note the location of the m-file because the path will be required in the PSCAD component.

5. The PSCAD module will be a custom component that is built by the user. Create a new component with two inputs and two outputs.
    a. Go to view and check in the component wizard
    b. Name it “Math” and in Line 1 put “MATLAB” and in Line 2 “Component”
    c. There will be 2 connections on the Left and 2 on the Right



    d. The connections will look like the following as they are brought up in the wizard, set the “Display Label” to be the same as the “Connection Name”.

Connection Name	Display Label	Connection Type	Data Type
in1	in1	Input Data	Real

in2	in2	Input Data	Real
out1	out1	Output Data	Real
out2	out2	Output Data	Real

e. Once the wizard is finished, place the block on the circuit page

6. Right click on the component and select “Edit Definition”. On the tab menu below the circuit space, select the “Parameters” tab.

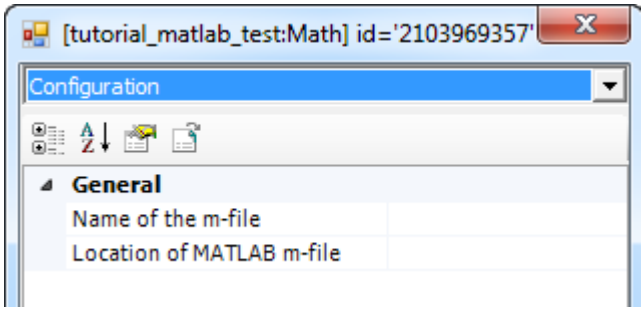
7. Create a new category by selecting the button () in the top menu by the dropdown box. Leave the name as the default and press “OK”

8. Right click in the grey area and select “New Parameter > Text Field”, place the box near the top of the grey area. Alternatively, click on the 

button on the menu.

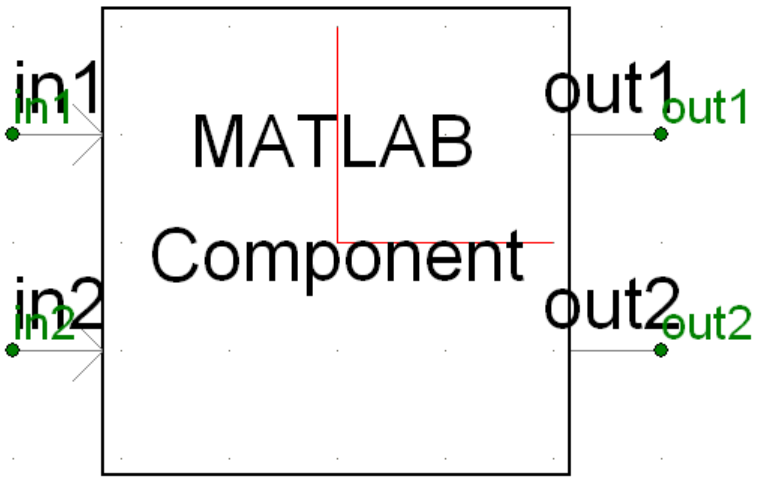
9. Double click the box that was placed and enter the “Symbol” value as *Name* and the “Description” as *Name of the m-file*.

10. Insert another “Text Field” and enter the “Symbol” value as *Path* and the “Description” as *Location of MATLAB m-file*.



(Note: The input area for each box may be adjusted by double-clicking on a parameter box and changing the “Width” value, the picture above uses 15 and 30. The same can be done with the entirety of the grey area by clicking the  button in the menu)

11. Go to the “Graphic” tab and double check to make sure you have two nodes on the left, one being *in1* and the other *in2* and on the right there should be *out1* and *out2*. If not, fix this before continuing. If all the nodes are present and correct, select the “Script” tab.



12. The Fortran code to be placed in the “Script” tab will look as follows:

```
#STORAGE REAL:4
! ------------------------------------------------
! ------------------------------------------------
! Input
      STORF(NSTORF) = $in1
      STORF(NSTORF+1) = $in2
!
! Call MATLAB Command
      CALL MLAB_INT("$Path", "$Name", "R R", "R R")
!
! Outputs from MATLAB
      $out1 = STORF(NSTORF+2)
      $out2 = STORF(NSTORF+3)
!
! Increment STOR* Pointers
      NSTORF = NSTORF + 4
! ------------------------------------------------
! ------------------------------------------------
```

13. Let’s look at each line of code…

```
#STORAGE REAL:4
```

The #STORAGE directive designates to the program locations to store values. In this case, storage is necessary for two inputs being sent to MATLAB and two outputs being received from MATLAB. That brings the total storage to four. All of these values are real.

```
STORF(NSTORF) = $in1
STORF(NSTORF+1) = $in2
```

The command STORF stores the value of the variable *in1* into the location given by the pointer NSTORF. The "$" is a substitution symbol. The same action occurs in the second line but a value of one must be added to NSTORF to give the variable *in2* a different location to be saved. If the plus one is omitted, the value of *in2* will overwrite the value of *in1* in the location designated by NSTORF, therefore losing the value of *in1* in memory.

```
CALL MLAB_INT("$Path", "$Name", "R R", "R R")
```

In order to initialize MATLAB, the command above is used. The *Path* line corresponds to the text input of the component when selected and tells the command where to find the m-file on the user's hard drive. The *Name* line is also a component text input and is the name of the m-file. Again, these two variables, *Path* and *Name*, may be changed to any name and can be done in the "Parameters" tab in their corresponding text input controls. The last portion of the line, … *"R R", "R R")*, tells the command there are two real inputs and two real outputs. This is the reason for using "R". If any were integers, an "I" would be placed instead of an "R".

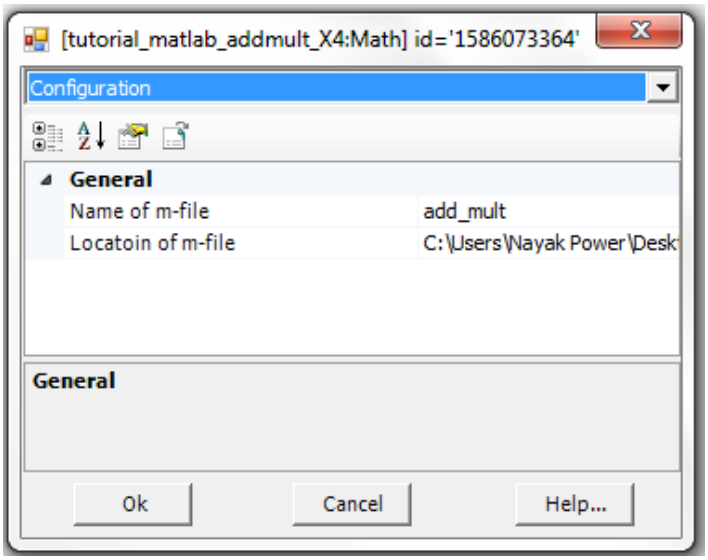
```
$out1 = STORF(NSTORF+2)
$out2 = STORF(NSTORF+3)
```

These two lines take the output from the MATLAB script and set them as outputs to the component block so they may be used by PSCAD. The names *out1* and *out2* were given to the output nodes at the beginning of the tutorial and can be named anything as long as it is changed in the script as well as in the "Graphic" tab so they match. The outputs are stored after the inputs in memory and that is why a +2 and +3 are necessary to attain them.

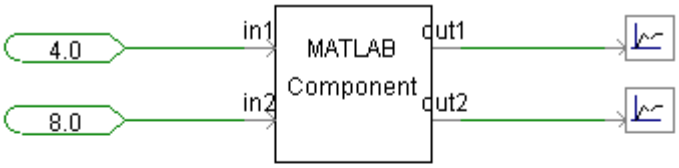
```
NSTORF = NSTORF + 4
```

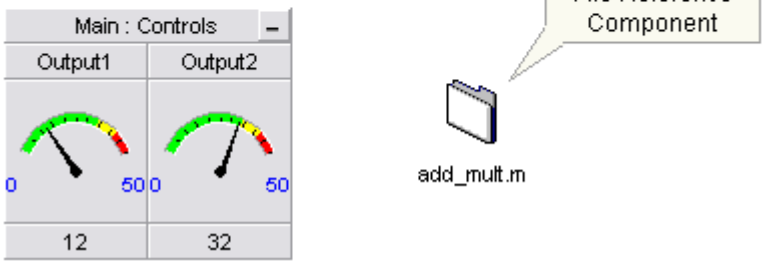
Lastly, the NSTORF pointer must be incremented in order to tell PSCAD that these locations in memory are already occupied by different values.

14. Double-click on the MATLAB Component and enter the parameters in their respective boxes.



15. Final results and circuit should resemble the picture below,

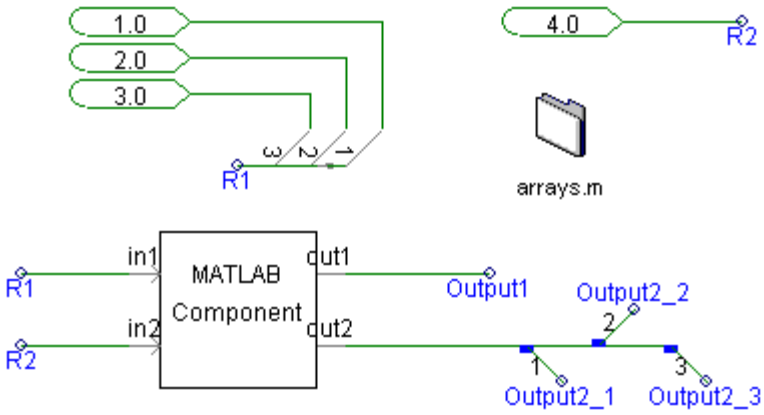




Tutorial 2

Using arrays with the MATLAB component

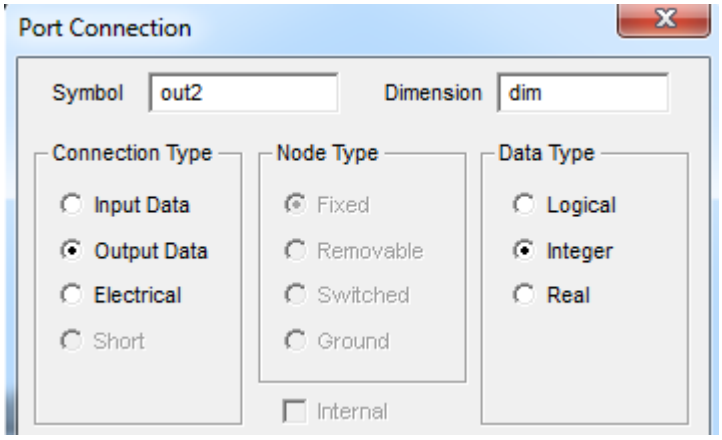
1. Take the existing case and save as with a new name (ex: tutorial_matlab_array.pscx). The final circuit will look like the one depicted below,



2. Select the MATLAB Component block and select the “Graphic” tab
    a. Change the dimension of each input *in1* and *in2* to 0. This will tell PSCAD that the dimension will be inherited based on the size of “R1” and therefore will not have to be constantly changed in the “Graphic” tab depending on how many inputs are desired. The outputs however must have an actual dimension value. In this case, the array output must be set to a value of 3. (Note: Another way to do this is to create a text input in the “Parameters” tab and use this variable as the dimension for the input on the “Graphic” tab.)

Location of MATLAB m-file	
Name of the m-file	
Top Output Dimension	1

untitled	Integer
Description	**Top Output Dimension**
Symbol	**dim1**
Group label	
Minimum value	**0**
Maximum value	
Data type	Literal
Intent	Input
Prompt text	
Help mode	Append
Conditional expression	
Default value	**1**



3. Select the "Script" tab in the "Edit Definition…" option of the component
    a. The script following is desired to handle the input/output arrays from the circuit that was created on Step 1,

```
#STORAGE REAL:8
#LOCAL INTEGER I_CNT, O_CNT
!-------------------------------------------------------------
!-------------------------------------------------------------
! Input arrays in1 and in2
      DO I_CNT = 1,$#DIM(in1),1
        STORF(NSTORF+I_CNT-1) = $in1(I_CNT)
      ENDDO
!
      STORF(NSTORF+3) = $in2
!
!Call MATLAB Command
      CALL MLAB_INT("$Path", "$Name", "R(3) R", "R R(3)")
!
! Outputs from MATLAB
      $out1 = STORF(NSTORF+4)
!
      DO O_CNT = 1,$dim,1
        $out2(O_CNT) = STORF(NSTORF+(O_CNT-1)+5)
      ENDDO
!
! Increment STORF* Pointers
      NSTORF = NSTORF + 8
!-------------------------------------------------------------
!-------------------------------------------------------------
```

4. Let's take a look at the new code additions…

```
#STORAGE REAL:8
#LOCAL INTEGER I_CNT, O_CNT
```

The #STORAGE directive is altered to match that of each cell for the two, three cell input arrays and the two, single cell outputs (3+1+1+3=8). The #LOCAL directive is used to initialize local variables and the *I_CNT* and *O_CNT* is used as a counter in the DO loop and going through each cell of the different arrays.

```
DO I_CNT = 1,$#DIM(in1),1
  STORF(NSTORF+I CNT-1) = $in1(I CNT)
ENDDO
```

The DO loop starts at one and counts sequentially to the value determined by the statement *$#DIM(in1)*. Simply stated, this command finds the dimension of the variable in parenthesis and eliminates the need to change the dimension of the input if it is desired to be altered to a different size at a later time. The loop itself is simply taking each cell of the array in variable *in1* and placing it into storage/memory.

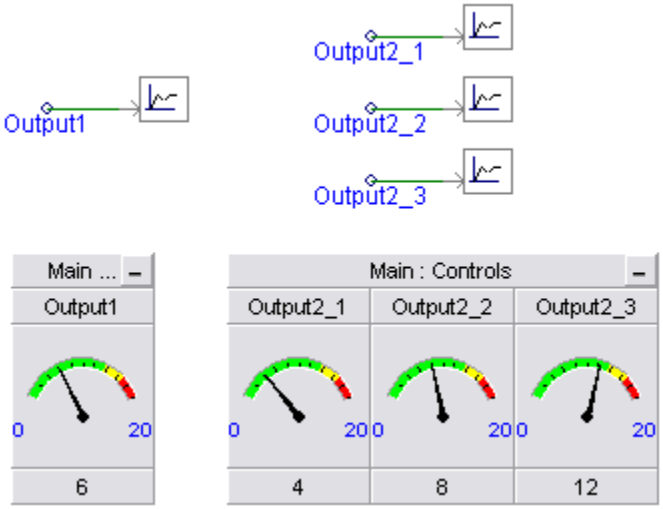
```
CALL MLAB_INT("$Path", "$Name", "R(3) R", "R R(3)")
```

The command to call the MATLAB environment is very similar to the one used prior but the inputs must have the addition of the dimensions for each corresponding variable.

5. Any MATLAB script can be used. The code below was used to verify the component is functioning properly between PSCAD and MATLAB before performing any complex operations with arrays,

```
1   function [out1, out2] = arrays(in1, in2)
2
3 -   out1 = in1(1) + in1(2) + in1(3);
4 -   out2 = in2*in1;
```

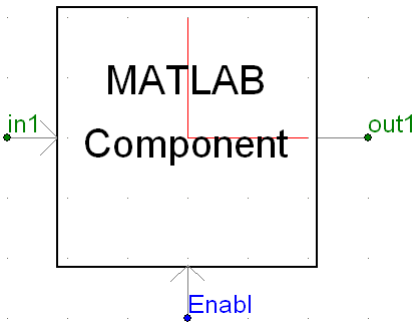
6. The output should be the following result…

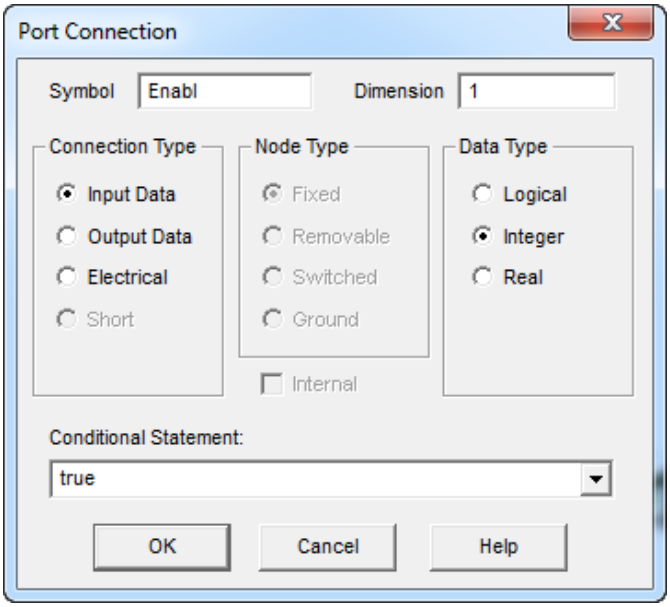


Tutorial 3

Implementing an Enable Switch for the MATLAB component

1. It may be desired to not have the MATLAB script run every time step so having some sort of enable/disable pulse or threshold parameter will speed up the overall simulation. Start by creating the graphic shown below in a new PSCAD case with the "Enabl" node set as an integer data type.



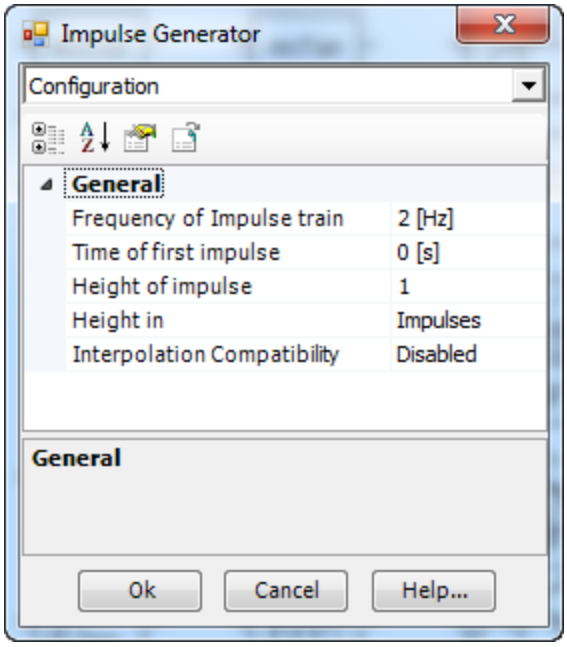


2. The FORTRAN necessary for the enable/disable functionality is implemented in a simple way with an IF statement surrounding the entirety of the code contained in the MATLAB Component block. An example is provided below,

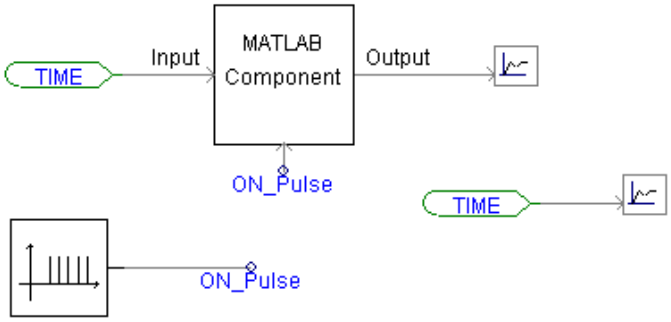
```
#STORAGE REAL:2
!-----------------------------------------------------
!-----------------------------------------------------
! IF statement to determine whether to run the script
! base on the value of the Enabl impulse
     IF ($Enabl.GT.0.9) THEN
!
! Pass inputs and outputs to and from MATLAB m-file
     STORF(NSTORF) = $in1
     CALL MLAB_INT("$Path", "$Name", "R", "R")
     $out1 = STORF(NSTORF+1)
!
     ENDIF
!
! Increment the NSTOR* pointer
     NSTORF = NSTORF + 2
!-----------------------------------------------------
!-----------------------------------------------------
```

3. Add the "Impulse Generator" component from the CSMF category in the master library with the parameters listed below. The circuit drawing should look similar to that of the picture shown

below as well. The 2000 Hz frequency is used based on a 250 µs Channel plot step. That is $(1/250*10^{-6})/2$. The MATLAB component will output a result every other time.



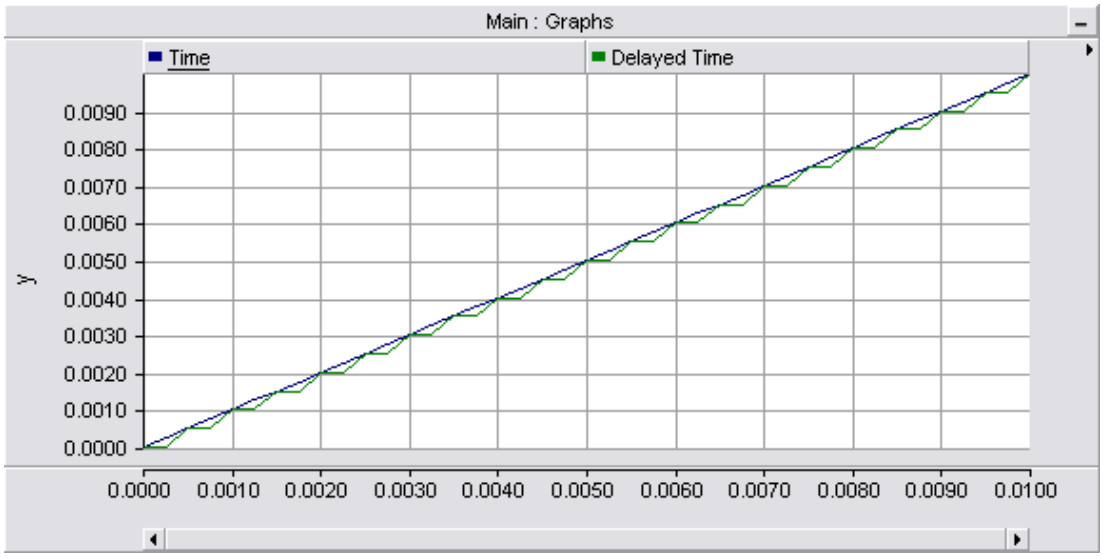
The final circuit is shown below,

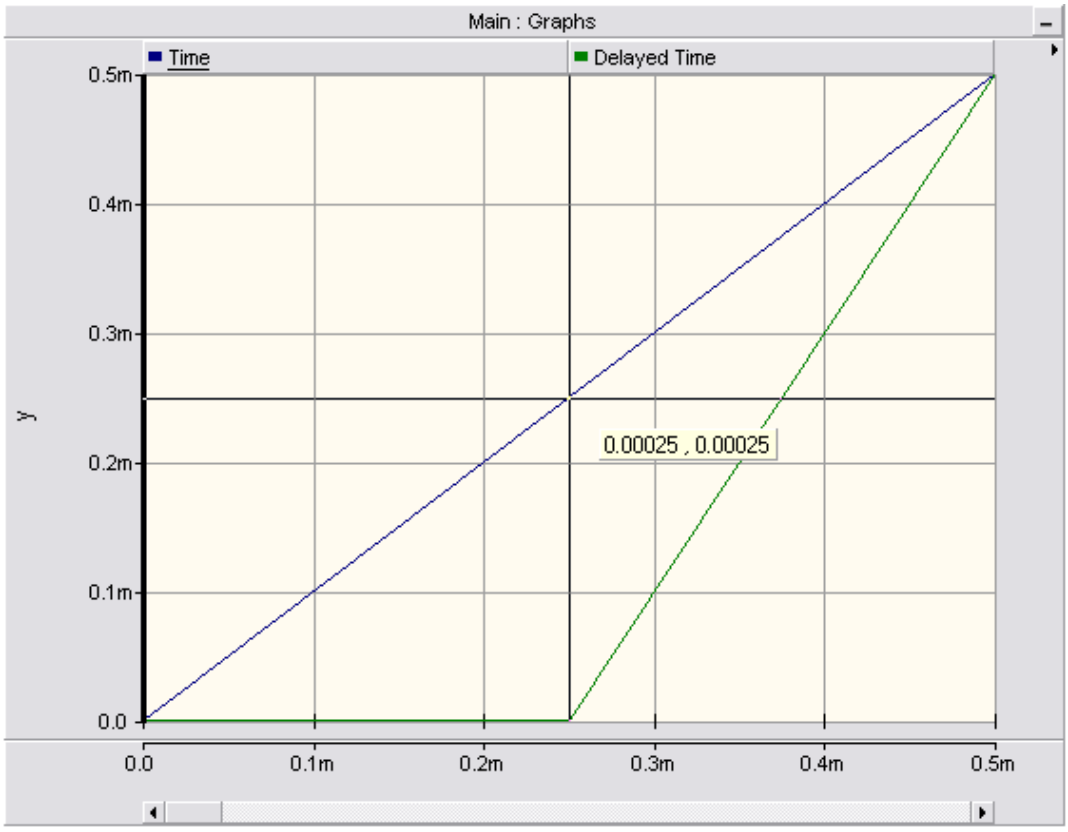


4. To make the analysis very simple and straight forward, create a MATLAB script that simply takes the input and forwards it to the output as depicted below,

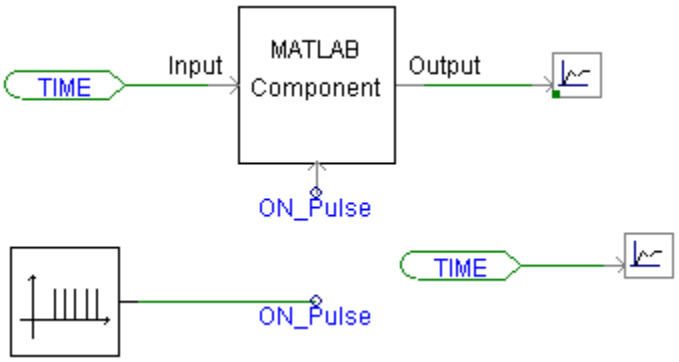
```
1    function out = enable_test(in)
2
3 -      out = in;
```

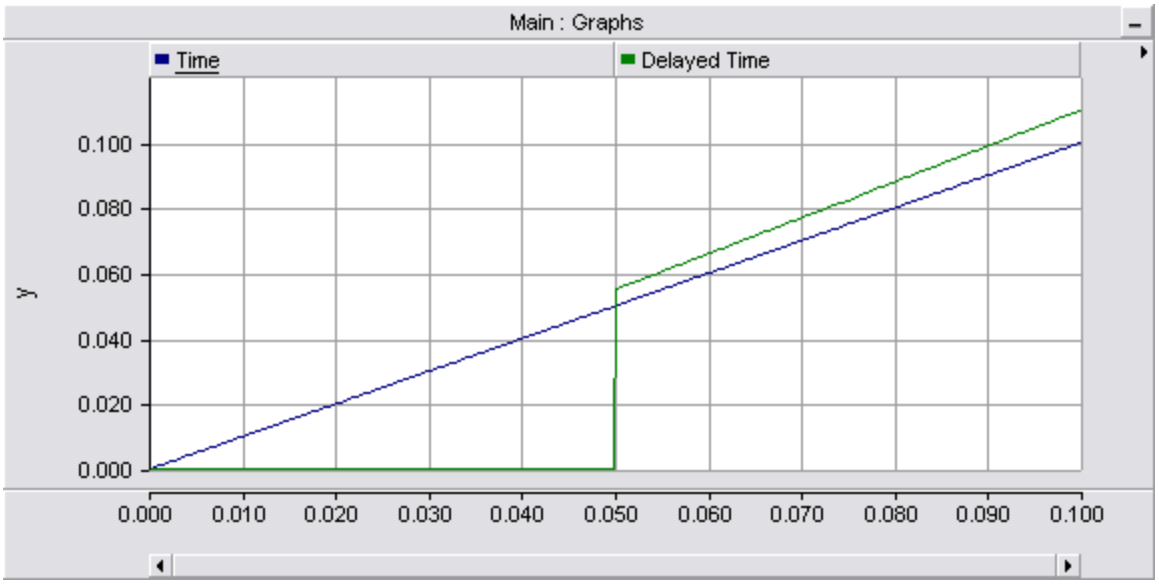
5. Zooming in on the graph created from the two time values, the results should appear as displayed below. It can be seen that when compared to the constant time output of the "Time" component block, the MATLAB does not have an output at every channel plot step. In this case, the channel plot is left at the default of 250 µs. At the crosshair there is no new plot point from MATLAB. It does changes values on the second channel plot of 500 µs though. Run for 0.01 sec.





6. By adjusting the frequency of the impulse, the MATLAB component can be set to run every few times rather than at every time step. By setting the “Pulse Generator” to not start until 0.05 seconds and altering the pulse to 4000 Hz, it can be visually seen how running the MATLAB block affects the total simulation time when it starts halfway through. Change the “Scale Factor” in the “Output Channel“ component coming from the output of the MATLAB block to 1.1 so that the two plot lines do not appear one of top of the other. Run for 0.1 sec.



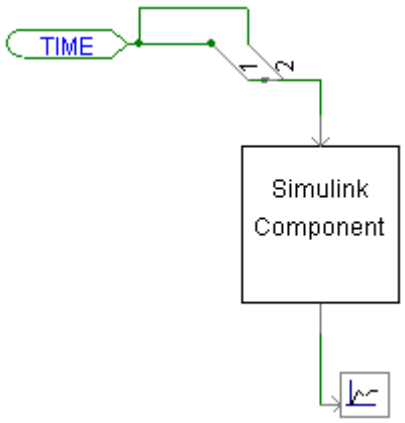


As an added note on simulation time, running a case that simply outputs the same input of a MATLAB component for a 1 second simulation time, 50 µs time step, and 250 µs channel plot step takes about 129 seconds to run from start to finish.

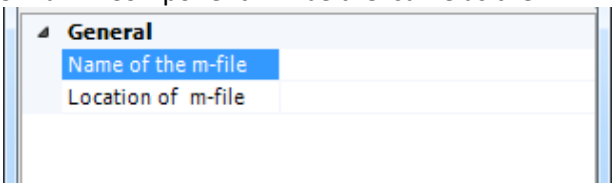
Tutorial 4

MATLAB Simulink Component

1. Create a module with a single input and single output, both with data type “Real”. The dimension of the input will be two. The input will be tied to a time block. The reason for placing the time value in both cells of the array is that Simulink only accepts a minimum of a two-dimensional array and the first cell is not easily accessible. Therefore, for all intents and purposes, the first cell of any input array may be set to any value since it will be acting as a “dummy input” to satisfy the two-dimension requirement.



The “Parameters” for the Simulink Component will be the same as the MATLAB component,



General	
Name of the m-file	Aa Text
Description	Name of the m-file
Symbol	Name
Default value	
Group label	
Prompt text	
Help mode	Append
Regular expression	
Error text	
Conditional expression	
Location of m-file	Aa Text
Description	Location of m-file
Symbol	Path
Default value	
Group label	
Prompt text	
Help mode	Append
Regular expression	
Error text	
Conditional expression	

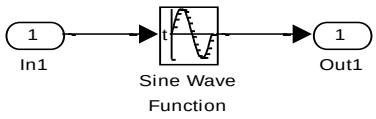
2. The FORTRAN script is as follows,

```
1  #STORAGE REAL:3
2  !
3  ! Store the input array in memory
4       STORF(NSTORF) = $in(1)
5       STORF(NSTORF+1) = $in(2)
6  !
7  ! Initiate Simulink
8       CALL SIMULINK_INT("$Path", "$Name", "R(2)")
9  !
10 ! Grab the outputs of the Simulink model from memory
11      $out = STORF(NSTORF+2)
12 !
13 ! Update the NSTORF pointer
14      NSTORF = NSTORF + 3
```

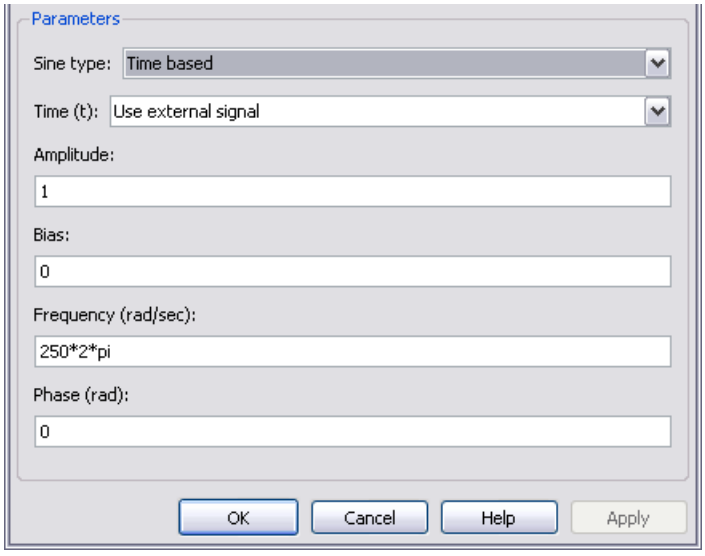
```
CALL SIMULINK_INT("$Path", "$Name", "R(2)")
```

The only notable syntax difference for the Simulink command is the lack of having to provide a specific output parameter. In this example, there is only a single output but many more may be added by altering the STORAGE and NSTORF values appropriately to match the change (i.e. if sixteen outputs are wanted, the #STORAGE and NSTORF pointer will need to have values of 18 and "+ 18", respectively). There, of course, will need to be sixteen corresponding outputs in the Simulink model being utilized.

3. The Simulink model in MATLAB will be an in, sine, and out block.

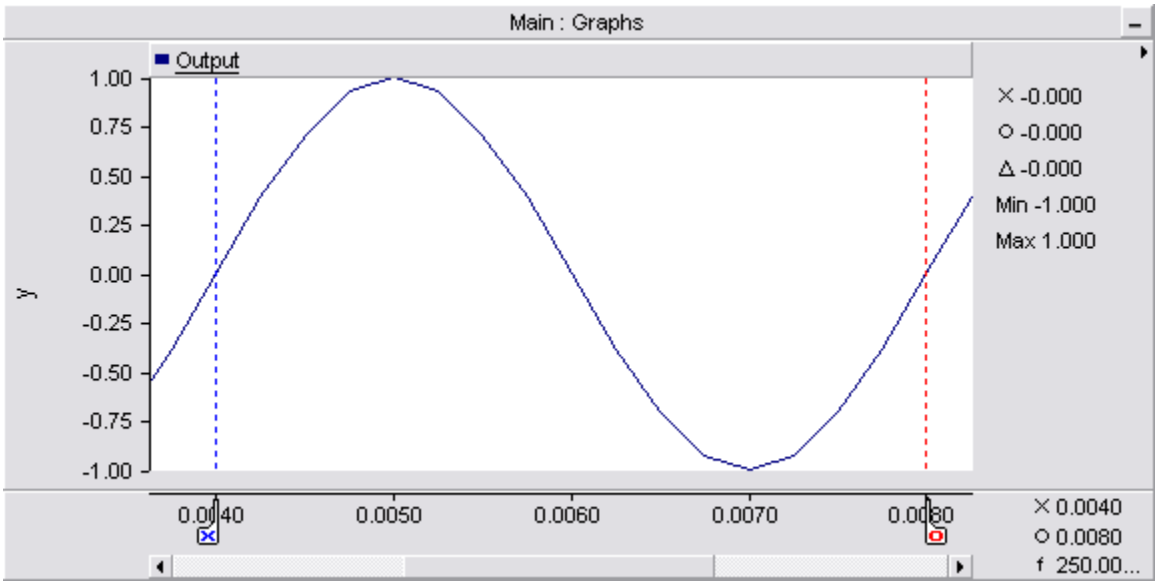


The parameters for the "Sine Wave Function" are,



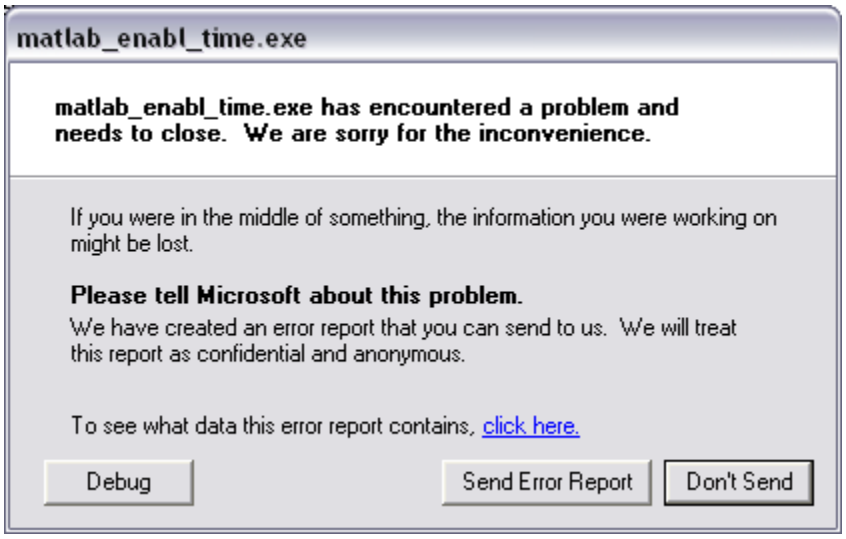
4. Set the simulation run time to match that of the value selected for the “Solution time step” in PSCAD. For this example, set both times to 250 µs so that a longer simulation time may be used to see the results clearly. Set the simulation duration to between 0.02 and 0.05 second.

5. The output should be similar to the results shown in the following plot. Note the frequency value in the lower right corner of the graph window.

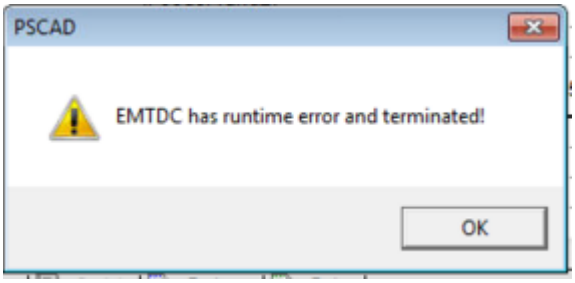


Troubleshooting

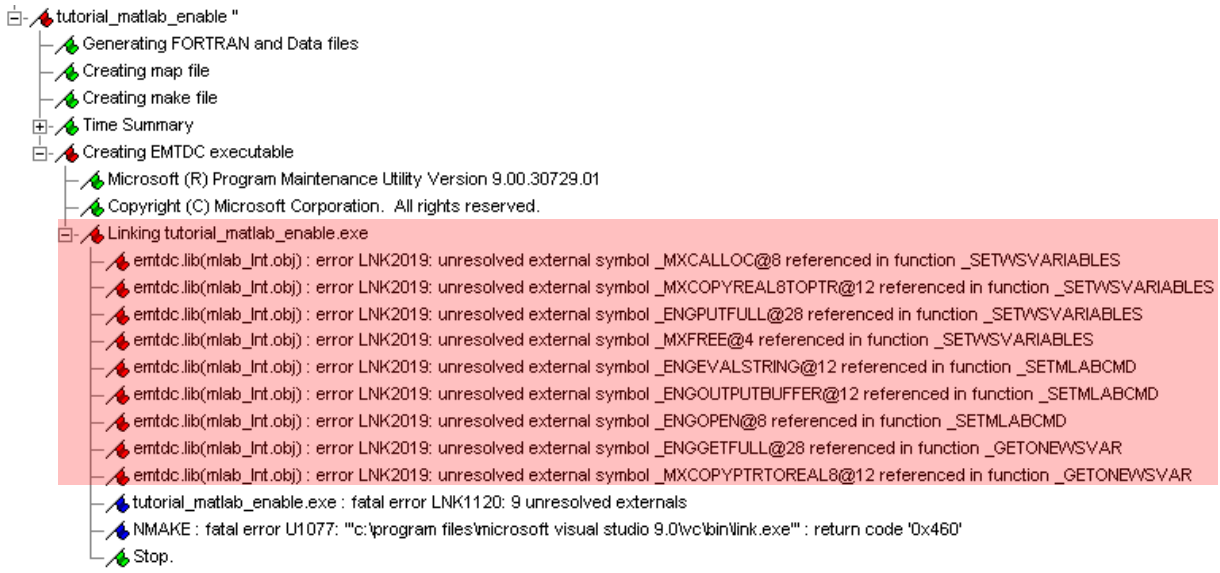
- If an error appears like the following, where “matlab_enabl_time.exe” will be replaced with the file name of your current PSCAD file, check to be sure the MATLAB function name in the component block matches with that of the actual m-file. A misspelling will result in PSCAD not being able to find the corresponding MATLAB m-file.



- The errors below will appear one and then the other if your file path pointing to the m-file is incorrect.



- These errors will appear in the “Build” dialog at the bottom of PSCAD under the circuit window and indicate that the MATLAB libraries have not been linked to. To fix this, go to “Project Settings > Link” and check the box “Link this simulation with the currently installed Matlab libraries”.



- The error “Generating FORTRAN and Data files” refers to something in the MATLAB component that is incorrect. A search throughout the component, code, variables, etc. will fix the error. If multilple custom blocks are present in the PSCAD case, the Fortran error may be present in a different module. (Note: In the error listed, the variable *out* does not exist in the component and the error was caused by a simple typographical error. It should be fixed to read *out1*.)