IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)
e-ISSN: 2278-1676 Volume 4, Issue 6 (Mar. - Apr. 2013), PP 31-37
www.iosrjournals.org

Performance Of System Functions as Effective Simulation Tool For Implementation Of Customized Algorithms

P.Meena [1], Dr. K. UmaRao [2], Dr. Ravishankar Deekshit [3]
[1](EEE, B.M.S. College of Engineering, India)
[2](EEE, R.V. College of Engineering, India)
[3](EEE,B.M.S. College of Engineering, India)

Abstract: *This work presents the results obtained by using System Functions (S functions) and comparing it with that obtained using Matlab Code Functions to customize an algorithm. The algorithm developed is to detect short duration voltage disturbances in power supply systems. These disturbances cause deviations in voltage current and frequency from their nominal value which eventually results in failure or mis-operation of sensitive equipment connected to the supply. The efficient detection, record and mitigation of these disturbances is the prime focus of research in the area of power quality improvement. Hence quickness and accuracy associated with the detection process are vital parameters that govern the effectiveness of these detection algorithms. The customized algorithm is intended to be implemented on a Digital Signal Processor through an integrated Matlab/ Simulink environment. The measure of effectiveness of the methodology of implementation facilitates an optimal choice. The paper presents the results obtained using System Function for simulation of the detection algorithm and comparison with that obtained by using Matlab Code Function.*

Keywords: *S-functions,Matlab-EmbeddedFunctions,effectiveness,accuracy,algorithm .*

I. INTRODUCTION

The S-functions (system-functions) provide a powerful mechanism for extending the capabilities of Simulink. An S-function is a computer language description of a Simulink block [1]. S-functions can be written in MATLAB, C, C++, Ada, or Fortran. C, C++, Ada, and Fortran S-functions are compiled as MEX-files using the mex utility.

The most common use of S-functions is to create custom Simulink blocks. We can use S-functions for a variety of applications, including, adding new general purpose blocks to Simulink, adding blocks that represent hardware device drivers, Incorporating existing C code into a simulation, describing a system as a set of mathematical equations and using graphical animations. An advantage of using S-functions is that, we can build a general-purpose block that we can use many times in a model, varying parameters with each instance of the block. Matlab also provides a user-defined function known as Matlab code functions which is a Matlab program that is created by the user, saved as a function file, and then used like a built -in function.

In this paper the system block developed is intended to detect short duration disturbances that occur in power supply systems in real time using a digital signal processor through an integrated Matlab/Simulink environment using the embedded link to code composer studio. The disturbances have duration ranging from half a cycle to one minute. The frequent occurrences of these disturbances do affect the performance and life of sensitive digital equipment and corrupt digital data. Quick detection of these disturbances is necessary for initiating restoration mechanisms that follow such as activation of a power line conditioner or a dynamic voltage restorer. The details of the simulation stages in simulink environment and how the function can be developed to customize any algorithm is initially discussed, followed by the description of the detection algorithm The results measuring the effectiveness of detection using the system function developed over that using Matlab code functions for the purpose are presented.

II. SIMULATION STAGES

Execution of a Simulink model proceeds in stages. First comes the initialization phase. In this phase, Simulink incorporates library blocks into the model, propagates widths, data types, and sample times, evaluates block parameters, determines block execution order, and allocates memory. Then Simulink enters a simulation loop, where each pass through the loop is referred to as a simulation step. During each simulation step, Simulink executes each of the model's blocks in the order determined during initialization. For each block, Simulink invokes functions that compute the block's states, derivatives, and outputs for the current sample time. This continues until the simulation is complete.

III. S-FUNCTION CALLBACK METHODS

An S-function comprises a set of S-function call-back methods that perform tasks required at each simulation stage. During simulation of a model, at each simulation stage, Simulink calls the appropriate methods for each S-Function block in the model. Tasks performed by S-function methods include ,

3.1. Initialization

Prior to the first simulation loop, Simulink initializes the S-function. In this stage, Simulink initializes the Sim Struct, a simulation structure that contains information about the S- function, Sets the number and dimensions of input and output ports. Sets the block sample times. allocates storage areas and the sizes array.

3.2.Calculation of next sample hit

If a variable sample time block is created, this stage calculates the time of the next sample hit; that is, it calculates the next step size.

3.3.Update of discrete states in the major time step

In this call, all blocks should perform once-per-time-step activities such as updating discrete states for next time around the simulation loop.

3.4.Integration

This applies to models with continuous states and/or non-sampled zero crossings. If S-function has continuous states, Simulink calls the output and derivative portions of S-function at minor time steps. So Simulink can compute the states for S-function. If S-function (C MEX only) has non-sampled zero crossings, Simulink calls the output and zero-crossings portions of S-function at minor time steps so that it can locate the zero crossings.

IV. USING S-FUNCTION IN MODELS

To incorporate an S-function into a Simulink model, the S-Function block is dragged from the Simulink User-Defined Functions block library into the model. The name of the S-function is specified in the S-function name field of the S-Function block's dialog box

V. S-FUNCTION CONCEPTS

Direct feed through means that the output (or the variable sample time for variable sample time blocks) is controlled directly by the value of an input port. A good rule of thumb is that an S-function input port has direct feed through if the outputs function is a function of the input u. That is, there is direct feed through if the input u is accessed in model outputs. The "time of next hit" function of a variable sample time S-function accesses the input u. An example of a system that requires its inputs (i.e., has direct feed through)is the operation $y=k*u$, where u is the input, k is the gain, and y is the output.

It is very important to set the direct feed through flag correctly because it affects the execution order of the blocks in the model and is used to detect algebraic loops.

Dynamically Sized Arrays-S-functions can be written to support arbitrary input dimensions. In this case, the actual input dimensions are determined dynamically when a simulation is started by evaluating the dimensions of the input vector driving the S-function. The input dimensions can also be used to determine the number of continuous states, the number of discrete states, and the number of outputs.

Setting Sample Times and Offsets-Both M-file and C MEX S-functions allow a high degree of flexibility in specifying when an S-function executes. Simulink provides the following options for sample times,

5.1.Continuous sample time

For S-functions that have continuous states and/or non-sampled zero crossings for explanation. For this type of S-function, the output changes in minor time steps.

5.2.Continuous but fixed in minor time step sample time

For S-functions that need to execute at every major simulation step, but do not change value during minor time steps.

5.3.Discrete sample time

If S-Function block's behavior is a function of discrete time intervals, you can define a sample time to control when Simulink calls the block. we can also define an offset that delays each sample time hit. The value of the offset cannot exceed the corresponding sample time. A sample time hit occurs at time values determined by the formula, $TimeHit = (n * period) + offset$. Where n, an integer, is the current simulation step. The first

value of n is always zero. If one defines a discrete sample time, Simulink calls the S-function model Output and model update routines at each sample time hit (as defined in the above equation).

5.4.Variable sample time

A discrete sample time where the intervals between sample hits can vary. At the start of each simulation step, S-functions with variable sample times are queried for the time of the next hit. Sometimes an S-Function block has no inherent sample time characteristics (that is, it is either continuous or discrete, depending on the sample time of some other block in the system).Then the block's sample time is specified as inherited

VI. WHY CHOOSE C FOR WRITING FUNCTION RATHER THAN MATLAB?

The set of callback methods, hence functionality, that C MEX-files can implement is much larger than that available for M-file S-functions. Unlike M-file S-functions, C MEX-files can access and modify the data structure that Simulink uses internally to store information about the S-function. This allows C MEX-files to implement a wider set of block features, such as the ability to handle matrix signals and multiple data types.
C MEX-file S-functions are required to implement only a small subset of the callback methods that Simulink defines. If the block does not implement a particular feature, such as matrix signals, we are free to omit the callback methods required to implement a feature. This allows one to create simple blocks very quickly.

VII. DIFFERENT APPROACHES FOR CREATING C MEX S-FUNCTION

The different approaches for creating C Mex S-Functions are,

7.1.Handcrafted S-Function

This can create S-function from scratch using a skeleton S-function (Template).

7.2.S-Function Builder

This block builds a C MEX S-function from specifications and code fragments that are supplied using a graphical interface. This eliminates the need to write S-functions from scratch.

7.3.Legacy Code Tool

This utility builds a C MEX S-function from existing C code and specifications that are supplied using MATLAB M-code.

Handcrafted S-function has been chosen in this work because although other two methods simplifies task of writing C-MEX S-functions they support fewer simulink features, whereas handcrafted s-function though difficult to write ,support widest range of simulink features.S-functions are written using an Application Program Interface (API) that allows to implement generic algorithms in the Simulink environment with a great deal of flexibility. This flexibility cannot always be maintained while using S-functions with Real-Time Workshop for example when implementing the algorithm with standalone devices like DSP. Although S-functions provide a generic and flexible solution for implementing complex algorithms in Simulink, the API incurs overhead in terms of memory and computation resources. In many cases in real-time embedded applications one can minimize memory and computational requirements by using the Target Language Compiler technology provided with Real-Time Workshop to inlining an S-function. One can inline S-function in two ways by, writing Wrapper S-function for the algorithm and callling those function using a wrapper S-function. An S function can also be in-lined by writing .tlc file for the algorithm. Though difficult to debug, accuracy of result is very high.

So by in lining an S-function, over head incurred by API and computational complexity can be eliminated. So speed of execution of algorithm on standalone devices like DSP is increased and accuracy of result is very high.

VIII. IMPLEMENTATION OF A DISTURBANCE DETECTION ALGORITHM USING S FUNCTION

Sags and swells are short duration voltage disturbances that occur in power supply systems. Sag is said to occur when the voltage falls between (0.1p.u-0.9p.u).Swell is said to occur when the voltage increases between(1.1p.u -1.9p.u.). The algorithm for sag and swell detection [2] is implemented using S-function and Matlab embedded function.

There are several detection methods for voltage sags in which sag voltages are usually expressed in terms of Root Mean Square (rms) values. Root Mean Square (rms) value of a signal can be used effectively to detect voltage sags and swells. Consider a periodic signal $v(t)$ of period T which is sampled at a frequency of f_s,the

sampling time being $T_s = \frac{1}{f_s}$ .The number of samples in one cycle is $M = \frac{T}{T_s}$ .Let $v(k)$ be the voltage at any sampling interval. For a discrete waveform, the rms values can be continuously calculated over a moving window of voltage samples ,the size of the window being M. Therefore, the rms value of the signal at any sampling interval is ,

$$V_{rms}(k) = \sqrt{\sum_{n=k-M}^{k} \frac{v^2(n)}{M}} \qquad (1)$$

During the occurrence of a disturbance say a voltage sag, the rms value drops below the nominal value. The drop is proportional to the level of sag. Similarly, during a voltage swell ,the rms value exceeds the nominal rms value by an amount proportional to the level of swell. If the rms value is calculated cumulatively, after the recovery of a sag or swell it takes a few cycles to come back to its nominal value. Thus there is a need to reset the rms calculation algorithm after the occurrence of sag or swell. The above problem can be overcome by calculating the rms value over a moving window encompassing a fixed number of samples M, as explained by the following equations.

$$S(k) = \sum_{n=0}^{M-1} v^2(k-n). \qquad (2)$$

Then,

$$S(k-1) = \sum_{n=0}^{M-1} v^2(k-(n-1)). \qquad (3)$$

The set of squared values of the samples obtained are updated by considering the latest sample as the last sample and discarding the oldest one.Let $(V_{rms}(k))$,indicate the rms values of the voltage signal at any sampling instant ***k*** evaluated as ,

$$V_{rms}(k) = \sqrt{\frac{S(k)}{M}} \qquad (4)$$

The rms values are found out and updated every sample time.. Thus the window and hence rms values are refreshed after every sample. The rms values at corresponding positions of two consecutive cycles are compared. $V_{rms}(k)$ is compared in magnitude with the value $V_{rms}(k+M)$ as shown in the equation below. Let the difference be ΔV_{rms} .Then,

$$\Delta V_{rms} = \frac{V_{rms}(k+M) - V_{rms}(k)}{V_{rms}(k)} \times 100 \qquad (5)$$

If the above difference is within a predetermined tolerance band the state of the system is declared as normal. On the other hand a trigger signal is generated when the difference is greater than a predetermined tolerance. If this value is >10%, then it is concluded that the voltage disturbance is a swell. If the above value is <-10%,it is concluded that the voltage disturbance is a sag. The polarity associated with the trigger signal also distinguishes the disturbance type. Positive polarity being indicative of a voltage sag and negative polarity indicative of voltage swell. After detection of the instant of occurrence of the disturbance it is important to detect when the system returns to normal state or in other words the recovery of the system from the disturbance. Monitoring the %change values for this may lead to erroneous results since a tolerance of 10% has been used for detection Therefore, the point of recovery from the disturbance is detected by monitoring the instant when the rms value returns to a value closest to the nominal value say 2-3% since in many cases it is seen that the grid voltage seldom reaches the nominal value.

IX. RESULTS

The algorithm for sag and swell detection is implemented in S function and also in embedded function. Results of S-function are found to be more accurate than the results of embedded function. The following results are useful in making a comparison between the two methods of algorithm development. The Figure.1.shows the input signal with sags and swells considered for test of the functioning of the algorithm using S Functions.

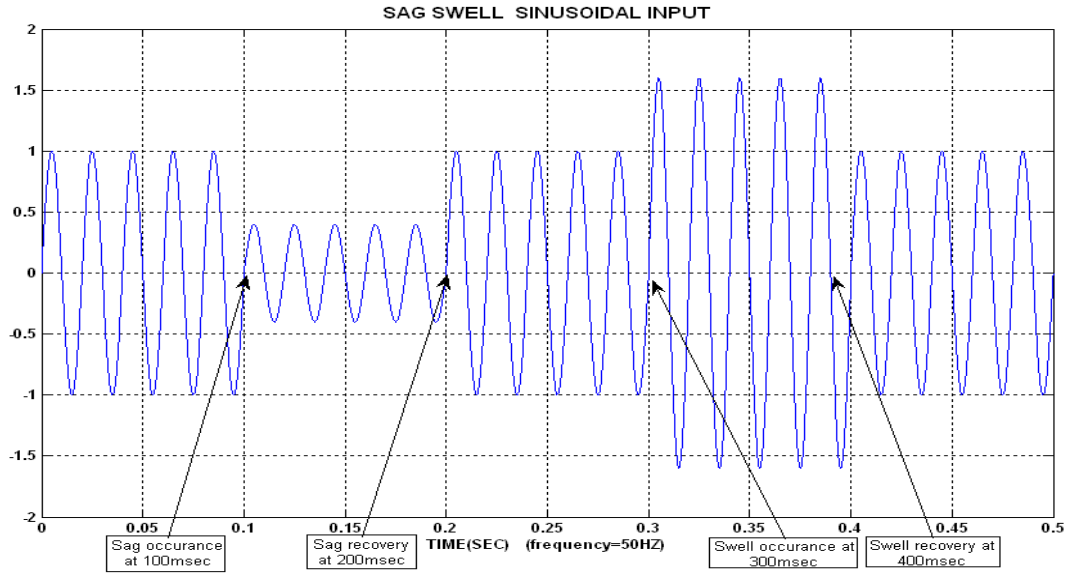

Figure.1.Input signal with Sags and Swells for test of the algorithm

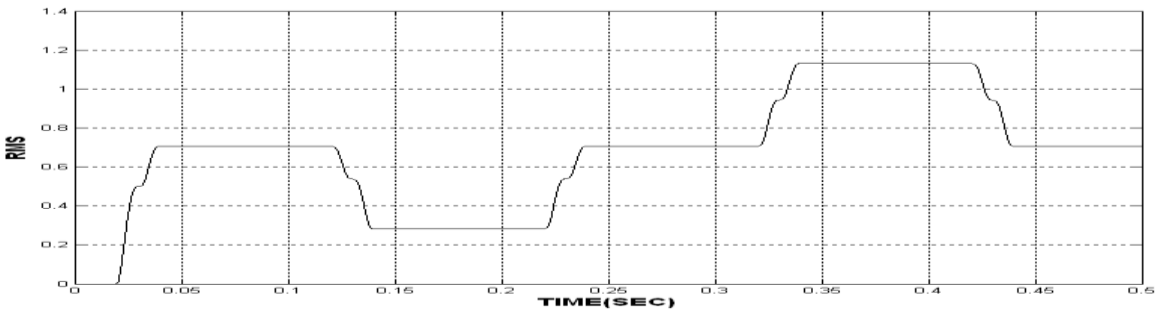

Figure.2.RMS plot obtained from using S Function

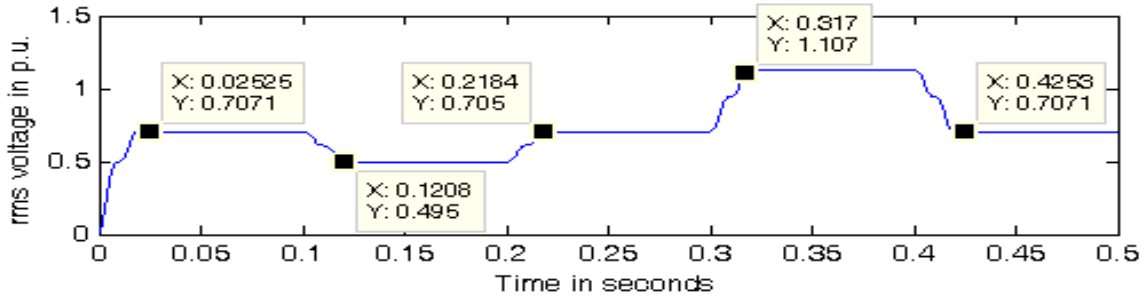

Figure.3.RMS plot obtained from using Matlab Code Function .

Figure.2 and Figure.3 shows the RMS variation plot obtained for the signal using the S Function and Matlab code Function for the detection algorithm. There is no appreciable variations observed in the magnitude of the rms values evaluated using the algorithm by the two functions . However, the variation in the rms values seem to be evaluated faster using S Functions as seen in Figure.2 as when compared to that shown using Matlab Code Functions in Figure. 3.

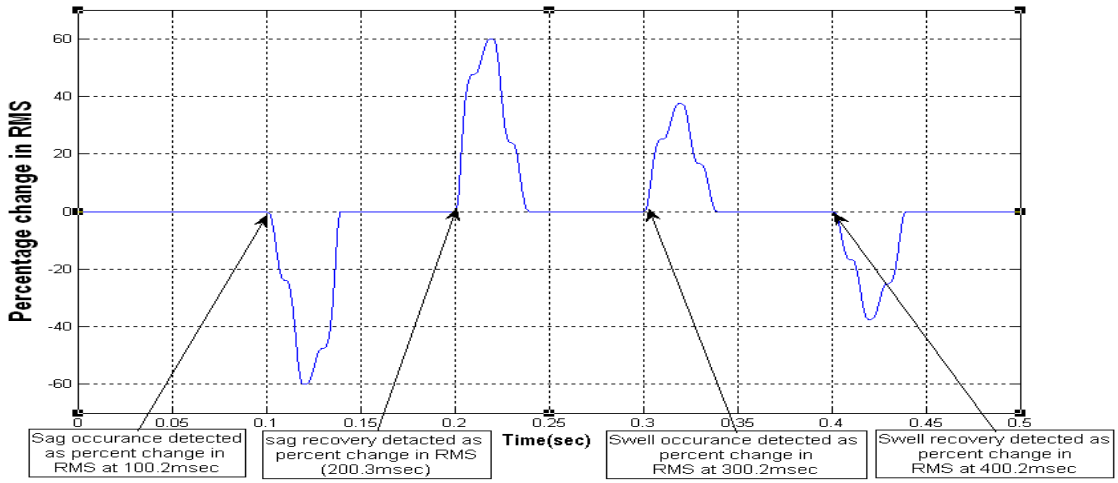

Figure.4.Percentage change in RMS plot using S Functions

Figures 4 and 5 show the percentage variation in the rms value evaluated for the signal shown in Figure.1 , using the Matlab Code Function and S Function for the detection algorithm. The results shown in Figures 6 and 7 show the trigger signal generated in response to the short duration voltage variations in the signal shown in Figure.1.From the values of time indicated in the Figures 6 and 7 of the trigger signal generated in response to the detection of sags and swells, it is clearly seen that, the detection algorithm implemented using S function in Matlab is simulated faster in operation than using Matlab code function.

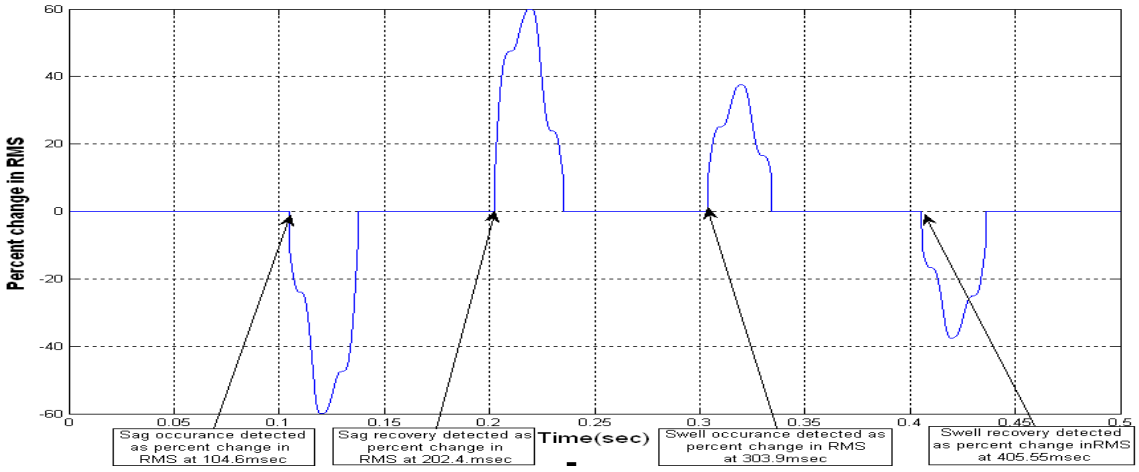

Figure.5. Percentage change in RMS plot using Matlab-code Function

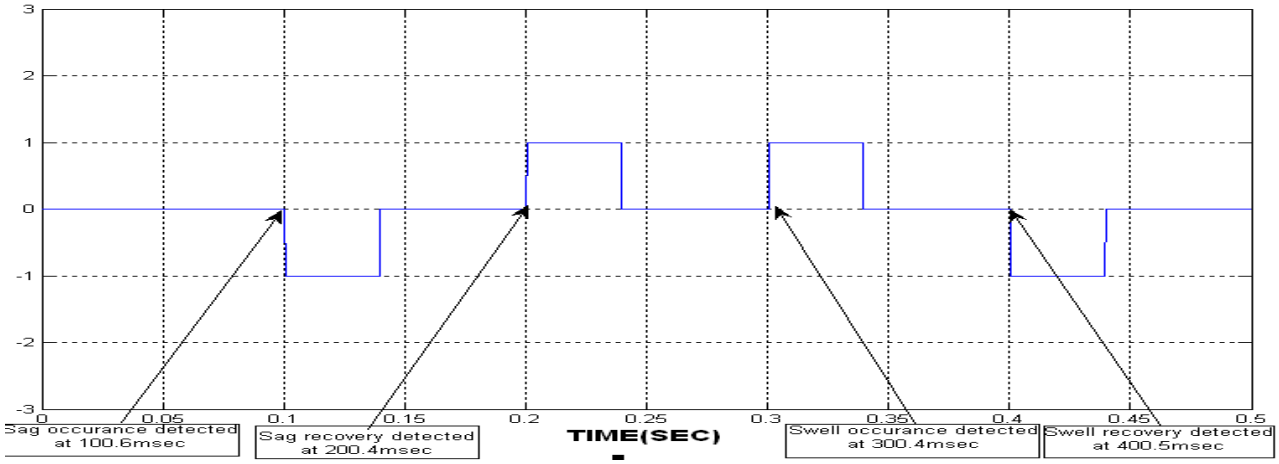

Figure.6.Trigger generated in response to detection of sag using Matlab-code Function

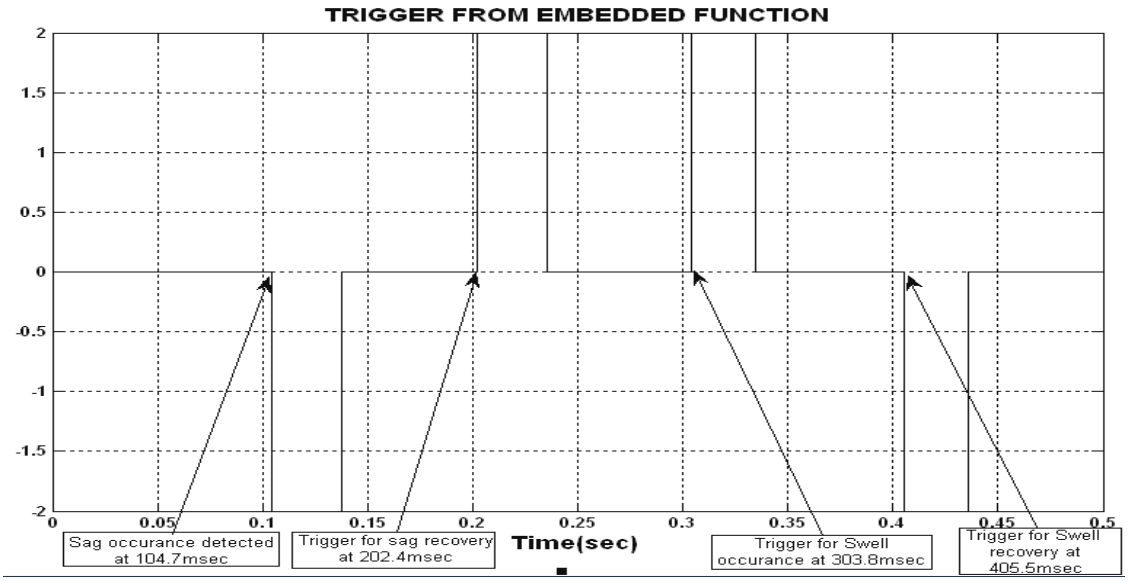

Figure.7. Trigger signal generated in response to sag detection using S Functions

X. CONCLUSION

The results obtained which are tabulated in the TABLE clearly distinguish the effectiveness of S function in quick disturbance detection when compared to use of Matlab embedded function. In situations where the algorithm is deployed on digital signal processing chips or other embedded processors this point is very crucial and useful.

TABLE

	SAG OCCURANCE	SAG RECOVERY	SWELL OCCURANCE	SWELL RECOVERY
ACTUAL TIME	**100ms**	**200ms**	**300ms**	**400ms**
EMBEDDED FUNCTION	**104.7ms**	**202.4ms**	**303.9ms**	**405.5msc**
S-FUNCTION	**100.6ms**	**200.4ms**	**300.4ms**	**400.5ms**

ACKNOWLEDGMENT

The authors acknowledge the contributions of Vittal. R to this work.

REFERENCES

[1]. "SIMULINK 7", *Writing S Functions by Math Works,www.mathworks.com*.
[2] ." A Simple Method For Real-Time Detection Of Voltage Sags and Swells in Practical Loads", *P.Meena,K.Uma Rao,Ravishankar Deekshit,EPE Journal 2011,vol.21,No.3,pp,1-8.*