

Adaptive Predictive Expert Control

---

# **ADEX Controller Toolbox for Simulink User Manual**

Edition: April 2008  
UM-TK-SM-01-EN

# Contents

About this Manual .....	3
Organization of this Manual .....	3
Conventions .....	3
Related Documentation .....	3
About ADEX Controller Toolbox for Simulink .....	4
Package Contents .....	4
System Requirements .....	4
Installing ADEX Controller Toolbox .....	4
ADEX Controller Toolbox Applications .....	6

## Part I

Integration of ADEX Controllers within Simulink .....	7
Introduction .....	7
ADEX Controller Toolbox Basic Features .....	7
ADEX Blocks .....	9
ADEX Controller .....	9
ADEX Reader .....	12
ADEX Writer .....	14
ADEX Parameter Browser .....	15

## Part II

Control Problem Simulated using the ADEX Controller Toolbox .....	18
Control Problem Description .....	18
Graphics and Simulation in the Simulink Environment .....	19

# About this Manual

This manual describes the ADEX Controller Toolbox for *Simulink* of *The Mathworks*. The Toolbox includes all the necessary tools for configuring and implementing ADEX controllers within *Simulink*.

## Organization of this Manual

The manual is divided into two main sections:

Part I – Integration of ADEX Controllers into *Simulink* - this chapter describes the ADEX blocks that are part of the Toolbox.

Part II - Developing an Example in *Simulink* – this section describes an example of a communication channel on which you want to remove noise by means of the definition of strategy based on the ADEX controllers.

## Conventions



This icon denotes a note, which notifies you of important information.



This icon denotes a warning, which warns you about information that should not be overlooked for the proper working of the system.

## Related Documentation

The following documents contain information that you might find helpful as you read this manual:

- *ADEX Methodology*.
- *ADEX Configurator*.

# About ADEX Controller Toolbox for Simulink

This section will list the contents of the ADEX Controller Toolbox for *Simulink*. It describes the installation steps and the applications of the ADEX Controller Toolbox.

## Package Contents

ADEX Controller Toolbox for *Simulink* contains the following:

- ADEX Controller Toolbox for *Simulink* CD.
- User manual of the ADEX Controller Toolbox for *Simulink*.
- Hardware Key (USB) that provides the necessary licenses to use ADEX systems.

## System Requirements

To install the ADEX Controller Toolbox for *Simulink*, the following are required as a minimum:

- Windows 2000/XP/Vista.
- *MATLAB* 7.0 or later.
- *Simulink* 6.0 or later.

## Installing ADEX Controller Toolbox

Complete the following steps to install the ADEX Controller Toolbox for *Simulink*:

1. Insert ADEX Controller Toolbox Installation CD.
2. Start a new session of *MATLAB* and establish the location of the area where the files of the package are installed as the Current Directory.
3. Execute the archive file *setup* either by clicking on the right hand button over the file within the Current Directory and pressing *Run* as shown in Figure 1 or writing *setup* in the command window of *MATLAB*. After a few seconds, if the installation has proceeded satisfactorily, an information window will appear otherwise an error message will be given.

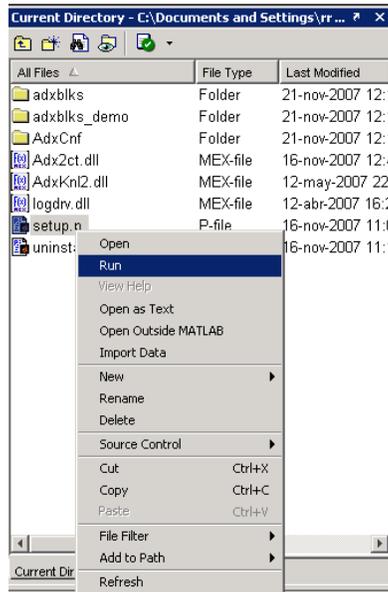


FIGURE 1. EXECUTING THE INSTALLER

4. Once the Toolbox has been installed, the installation package permits the installation of the ADEX COP 2 Configurator (see Figure 2).



FIGURE 2. WINDOW FOR INSTALLING THE CONFIGURATOR

5. If the Yes button is accepted, the installation of the ADEX COP 2 platform Configurator will begin. On completion, the drivers for the hardware security key will be launched. These are a requirement for running ADEX simulations, strategies and controllers. Finally, the system will prompt the user if it is also required to install LabVIEW Run-Time 7.1. This step can be bypassed if there is already a version of *LabVIEW 7.1* on the computer.

To **uninstall** the ADEX Controller Toolbox, the procedure is very similar to the steps listed above.

1. Start a new session of *MATLAB* and establish the location of the area where the files of the package are installed as the Current Directory.

2. Execute the archive *uninstall* either by clicking on the right hand button over the file within the Current Directory and pressing *Run* as shown in Figure 1 or writing *uninstall* in the command window of *MATLAB*.

It should be noted that, in this case, neither the equipment, *Configurator*, *USB security drivers*, nor *LabVIEW Run-Time 7.1* should be deleted. This process should be carried out from the computer Windows Control Panel using the option Add / Remove programs (Figure 3).



FIGURE 3. ADD/REMOVE PROGRAMS



After completing the installation, be sure to connect the Hardware key that contains licenses for the implementation of ADEX Systems before using the ADEX Controller Toolbox for *Simulink*.

## ADEX Controller Toolbox Applications

The purpose the Toolbox is to provide a means of simulating ADEX Systems in a familiar, known environment. This is the reason why the Adaptive Predictive Expert methodology has been integrated into *Simulink*, so that it can be used both for carrying out experiments/testing and creating industrial applications.

The ADEX Controller Toolbox for *Simulink* contains the necessary tools for developing control applications based on the ADEX methodology in this programming environment, allowing the inclusion of ADEX controllers in your schemes to generate systems from simple simulation models to industrial processes of great complexity.

## Part I

# Integration of ADEX Controllers within Simulink

## Introduction

ADEX Controller Toolbox enables ADEX controllers to be added to *Simulink* schemes for the design of control systems based on the ADEX methodology.

This kit is integrated into *Simulink* as an extra library within the user menu, so it is therefore convenient that the user obtains basic knowledge of this program prior to using the Toolbox. For more information on this environment, please visit the web site of *The Mathworks* at (<http://www.mathworks.com/>)

In the following sections, a full description is given of the Toolbox starting from how to install it to outlining features including the blocks which make it up. The 'Help' module of the Toolbox which is installed with *Simulink* includes a program allowing the user to access the ADEX parameters necessary for the configuration of the controller.

## ADEX Controller Toolbox Basic Features

Once the installation has been completed, the Toolbox will be available in the *Simulink Library Browser* as can be seen in Figure 4:

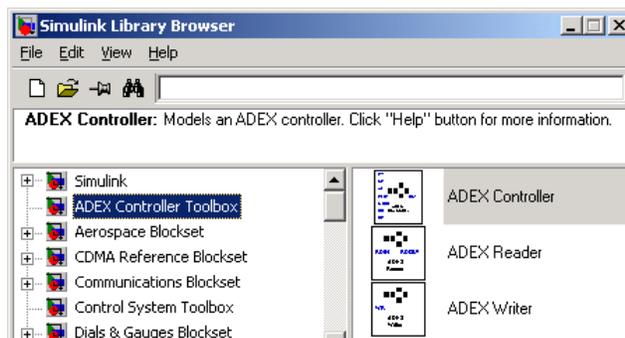


FIGURE 4. ADEX IN THE SIMULINK LIBRARY BROWSER

The ADEX Controller Toolbox comprises three elements which will be described in detail in the following sections:

- ADEX Controller block.
- ADEX Reader block.
- ADEX Writer block.

The most basic scheme, which can be implemented in *Simulink*, deals only with the first three blocks. The Reader and Writer blocks are optional, although they will always appear as options together with the group of blocks as listed above. Improvements in the system can be achieved with these, since their function is to manipulate controller configuration parameters. For example, they are useful for treating exceptions or limitations in the process; for those parameters in which it is desired to make changes before specific events occur.

To run a process under ADEX control, the system requires the following directories:

- **Directory Ctlrs:** In this, controller files relevant to the MDL models are stored inside the Current Directory. In each cycle, if there does not exist a controller with the desired name, one will be created in this directory and each time it is referenced, it will be found here. In addition, this directory will have a default controller called *ctlrcnf.dfl*.
- **Directory Logs:** Here will be stored a text archive whose purpose is to log information messages or errors generated by the controllers.

In any event, the Toolbox will check the existence of these Directories and create them automatically if they are missing at the start of the first simulation carried out in each *MATLAB* session. At these times, the Command Window will show some messages relating to this process. (Figures 5 & 6).

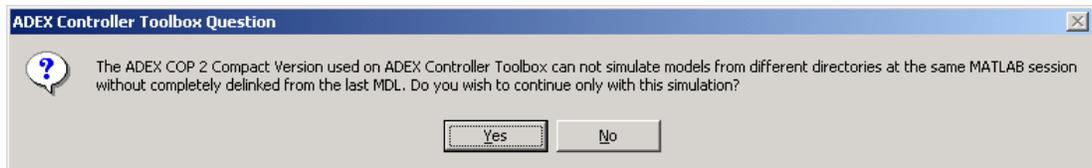
```
Warning(AdxCtrlr): Logs not found... Creating directory...
Info(AdxCtrlr): Logs OK
Warning(AdxCtrlr): Ctlrs not found... Creating directory...
Info(AdxCtrlr): Ctlrs OK
Warning(AdxCtrlr): Default File not found... Copying file...
Info(AdxCtrlr): Default File OK
```

**FIGURE 5. THE DIRECTORIES AND FILES DO NOT EXIST**

```
Info(AdxCtrlr): Logs OK
Info(AdxCtrlr): Ctlrs OK
Info(AdxCtrlr): Default File OK
```

**FIGURE 6. THE DIRECTORIES AND ARCHIVES DO EXIST**

The ADEX controllers can only be used when they are located in the same Current Directory. In addition, if there is a requirement to run two models consecutively, situated in different folders (without closing *MATLAB*), it is necessary to delete the references which the Toolbox created with the first scheme, and for this, a confirmation will be prompted as shown in Figure 7.



**FIGURE 7. CONFIRMATION OF CHANGE IN SIMULATION**

- If the response is “Yes”, the system will break the existing links with the previous model and proceed with the new one.
- If the response is “No”, the system will abort the execution change and it will be possible to return to running the previous scheme.

## ADEX Blocks

### ADEX Controller

As implied by the name, this module lies at the centre of ADEX systems since it is the actual controller. The module can be seen in Figure 8 below:

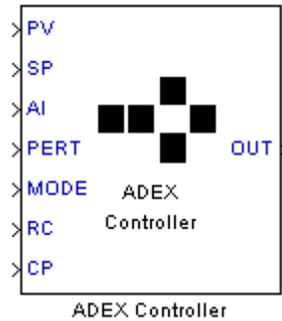


FIGURE 8. ADEX CONTROLLER

With respect to the input terminals:

- **SP (Set Point):** Value of the set point of the PV.
- **PV (Process Variable):** Process Output to the controller.
- **PERT (Perturbation):** Input available for the introduction of perturbations which affect the PV.
- **AI (Actual Input):** The actual value of the latest control action applied to the process. The values sent by the controller and the actual input do not necessarily have to be the same, for various limitations or for other reasons.
- **MODE:** The modes of operation of the controller.
- **RC (Rate of Change):** This is the maximum rate of change of the PV when it is approaching the set point.
- **CP (Control Period):** This is the time interval between control actions.

The output is as follows:

- **OUT:** Control action which must be applied to the process.

Double clicking on the diagram of the scheme, the mask of the block will appear (Figure 9). This represents the interface between the controller and the default values. In this way, the values characterizing the block are entered.

- **Controller Name:** This is the name used to create the controller and it will manipulate an archive CLR which stores the controller status in memory. In addition, it will identify the block for the other two modules as well as for ADEX COP 2 Configurator. Using *MATLAB*, it will be defined between single commas '...' (See Figure 9).

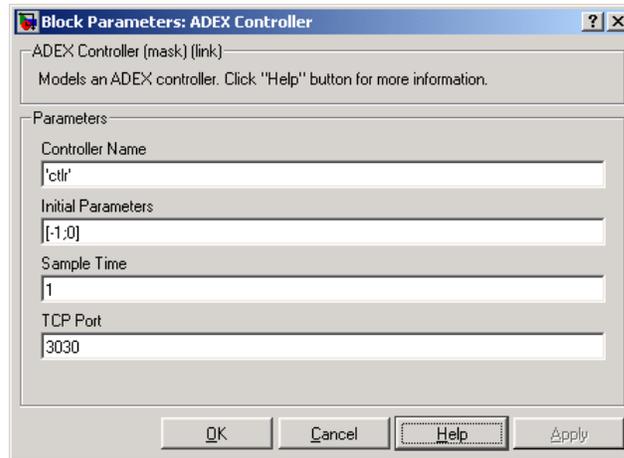


FIGURE 9. CONTROLLER MASK

- **Initial Parameters:** The values input here are those required when the controller execution starts. It is possible to modify those before and after the simulations as well as modify the name. The parameters are introduced by means of a 2 x N matrix, corresponding to the index and value pairs where N is the number of the parameter to be initialized.

index_1	.....	index_N
value_1	.....	value_N

This data can be entered in various ways:

- (i) Writing them manually in the *MATLAB* format:

[index\_1 index\_2 ..... index\_N; value\_1 value\_2 ..... value\_N]

- (ii) By means of a variable defined “by hand” in the Workspace following the same rules. In this box, the name of the variable must be written.
- (iii) The last option is using a variable called *Simulink.Parameter* with the matrix defined in the field *Value*. In this case, the name of this *Simulink.Parameter* type element can be written in the Workspace of *MATLAB*.

- **Sample Time:** This value must be that which corresponds to the parameter ST of the controller expressed in seconds. Since this is a module of *Simulink*, it is possible also to input, -1 to inherit it from the rest of the model.

- **TCP Port:** It's dedicated to communicate with the ADEX COP 2 Configurator.

## ADEX Reader

The ADEX Reader Block performs the function of reading the values of the parameters specified in the input terminal via its respective indices.

This block has the structure shown below (Figure 10):

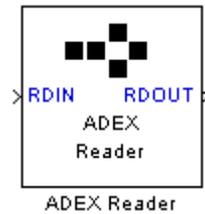


FIGURE 10. ADEX READER

- **RDIN:** This is the input to the module. This will be a vector of N elements, N being the number of parameters which have a value required to be known.



In a similar way to the controller, these indices can be input in a variety of ways.

- Writing them manually in the *Constant* block as follows:

[index\_1 index\_2 ..... index\_N]

- By means of lines coming from other blocks of *Simulink* which have double precision (for example not Boolean).
- By means of a variable defined “by hand” in the Workspace following the same rules. An instance of the module *From Workspace* is required.
- Another option is via a *Simulink.Parameter* variable with the vector defined in the field *Value*. It is also necessary in this case to include the block *From Workspace* from the *Sources* library in *Simulink*.

- **RDOU**T: This pin is the output from the module. It expresses the results of reading the parameter values specified in the input. Consequently, it is another vector with the same dimensions as the block input (for this case it will be N).

value_1	.....	value_N
---------	-------	---------

The values of the output are numeric type elements of double precision so that it can be connected to any other block which allows this type of inputs, including the modules Scope, Display, mathematical operations etc. Alternatively, connect them directly to the parameter writer which will be shown in the next section.

The next step is to explain the type of parameters and how they can be configured using the mask shown in Figure 11:

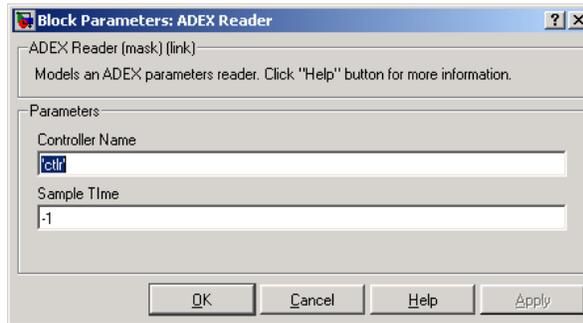


FIGURE 11. ADEX READER MASK

- **Controller Name:** The ADEX Reader needs to know the controller whose parameters need to be read. The user must ensure that this controller exists otherwise an error will be returned and the error code will be displayed in the Command Window of *MATLAB*. In the same way as for the ADEX Controller, the name must be enclosed in single commas '.....'.
- **Sample Time:** This value indicates the period of time in seconds which passes between two readings. It does not have to be the same as that of the controller action, as the timing of each parameter does not have the same significance. For example, if a process has a short response time, it is necessary to input an ST value to the controller which is of a similar magnitude so that the cause and effect mechanism works correctly. However, the user is at liberty to read parameters as frequently as is deemed necessary. Finally, with regard to 'inheritance', the value -1 can be input to this field.

## ADEX Writer

The objective of the third block of the Toolbox dedicated to adaptive predictive expert control is to modify the configuration parameters of the ADEX controller. These changes, as distinct from the controller module, are carried out during the cycles of the simulation.

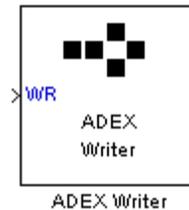


FIGURE 12. ADEX WRITER

The ADEX Writer element has only one input which is described as follows (see Figure 12).

- **WR:** The index-value pairs of the parameters to be changed appear in this terminal. Here, the input is the same as that for the initial controller parameters and is therefore also a 2 x N matrix, with N being the number of parameters to be modified.

index_1	.....	index_N
value_1	.....	value_N

This data can be input in various ways:

- (i) Writing them manually in the *Constant* block as follows:

[index\_1 index\_2 ..... index\_N; value\_1 value\_2 ..... value\_N]

- (ii) By means of lines coming from other blocks of *Simulink* which have double precision (for example not Boolean).
- (iii) By means of a variable defined “by hand” in the Workspace following the same rules. An instance of the module *From Workspace* is required.
- (iv) Another option is via a *Simulink.Parameter* variable with the vector defined in the field *Value*. It is also necessary in this case to include the block *From Workspace* from the *Sources* library in *Simulink*.

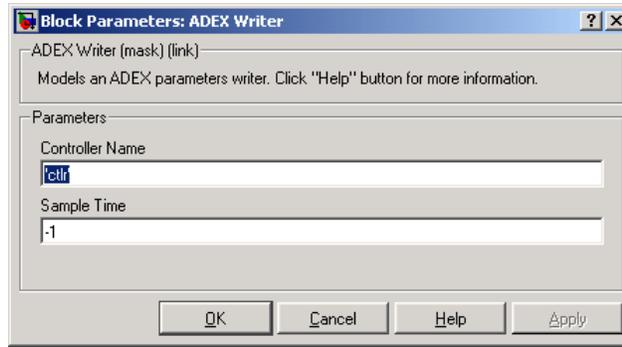


FIGURE 13. ADEX WRITER MASK

The mask which has the ADEX writer module will be described now (Figure 13):

- **Controller Name:** The ADEX Writer needs to know the controller whose parameters need to be write. The user must ensure that this controller exists otherwise an error will be returned and the error code will be displayed in the Command Window of *MATLAB*. The name must be enclosed in single commas '.....'.
- **Sample Time:** This value indicates the period of time in seconds which passes between two writings. It does not have to be the same as that of the controller action, as the timing of each parameter does not have the same significance. For example, if a process has a short response time, it is necessary to input an ST value to the controller which is of a similar magnitude so that the cause and effect mechanism works correctly. However, the user is at liberty to read parameters as frequently as is deemed necessary. Finally, with regard to 'inheritance', the value -1 can be input to this field.

## ADEX Parameter Browser

Having explained the functioning of the ADEX blocks, it is necessary to know the index of a parameter so that it can be read correctly. Using the name of the parameter in ADEX COP 2 Configurator, the indices and full names can be obtained by means of the ADEX parameters browser (Figure 14). This is referred to in plural since although a particular parameter is being searched, this could have several identifiers and names according to the domain it belongs to and the PV which it affects.

Moreover, in most cases, the key to be input coincides with the initials of the parameter in English (for example, the Sampling Time is ST).

It is possible to access the parameter search engine from the Help section of the Read and Write Parameter modules. The browser opens in a predetermined navigator window in the user's system (that is in Internet Explorer, Mozilla Firefox or similar) and has the following appearance:

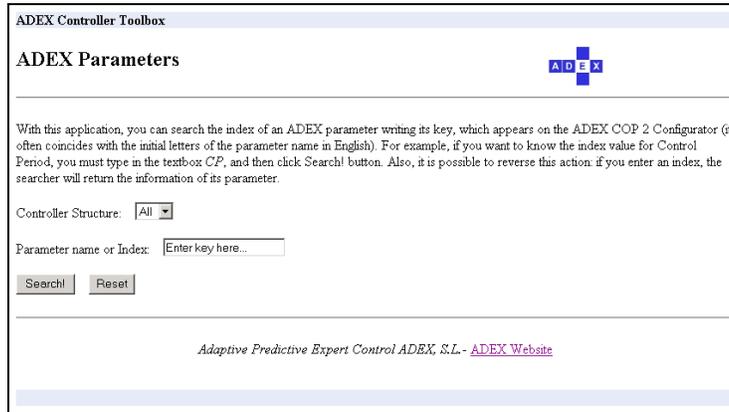


FIGURE 14. ADEX PARAMETER BROWSER

To simplify searches, the type of controller to be used can be specified, since the names and indices do not always coincide in the various structures. The types of controller are as follows:

- **1x1**: These are controllers with only one PV and no perturbations.
- **6x1**: These also have only one PV but up to 5 perturbations.
- **9x3**: This is typical of the remainder. In this case there are 3 PV's and up to 6 perturbations (PERTs).

For example, if we want to know the name of the parameter associated with the Prediction Horizon in a 6x1 driver, we should write *PH* in the textbox and select the appropriate structure, resulting in the following:

6x1					
ADEXCOP 2 Configurator key	Index	Parameter name	Process Var	Domain	Description
PH	26	Pvg1.Ap1.In	PV1	AP-L	Prediction horizon
PH	61	Pvg1.Ap2.In	PV1	AP-C	Prediction horizon
PH	96	Pvg1.Ap3.In	PV1	AP-U	Prediction horizon

FIGURE 15. SEARCH RESULTS

Another feature of the browser is that the inverse also works: if the index is input to the text box, the corresponding parameter will be found.

## Part II

# Control Problem Simulated using the ADEX Controller Toolbox

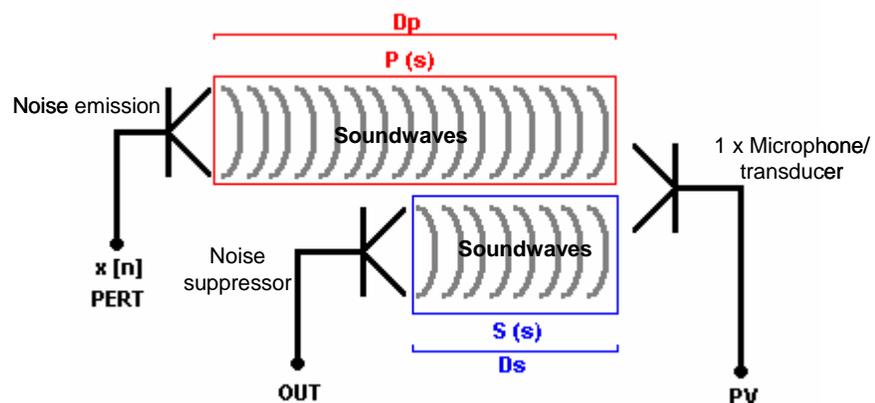
The purpose of this section is to illustrate how the Toolbox in *Simulink* can be implemented using a practical example. The objective is not to be a guide for configuring adaptive predictive expert controllers (although there are many references to the concepts), but rather, to familiarize the user with the toolbox.



You can access this demonstration through the ADEX Controller Toolbox section in *MATLAB* Help.

## Control Problem Description

The problem is centered on a communication channel where there is a need to minimize an acoustic noise. The noise is measured using a microphone or transducer which will allow it to be treated as a measurable perturbation. Since the objective of the exercise is to suppress this signal, the process variable needs to be zero as does the set point.



Interference between noise emission and noise suppressor aimed at making combined result zero volume. Control action aimed at changing suppressor amplitude to match emission and so eliminate

FIGURE 16. SCHEME OF THE SYSTEM

The role of the controller therefore is to generate a control action aimed at keeping the noise as close to zero as possible.

Figure 16 shows the overall scheme. The set point of the system is zero, there is a PERT signal which corresponds to the known noise and an OUT, calculated to eliminate the perturbation. The variable PV will be the sum of both effects: one caused by the noise and the other caused by the control action.

As can be seen from the diagram above, the emission of the OUT signal (noise suppressor) as far away from the noise emission as possible (where the PV can be read) to make it easier to anticipate the perturbation and obtain better results.

The distances between send and receive devices are modeled including the delays  $D_p$  and  $D_s$  such that the perturbation delay is greater than the control action delay. Given that the system is acoustic and the transmission medium is therefore air, the distance can be calculated according to the following equation:

$\text{Distance} = \text{Delay} \times \text{Speed of sound propagation}$
---

Finally, to complete this system in *Simulink*, the systems  $P(s)$  (noise) and  $S(s)$  (suppression) referred to as Primary and Secondary respectively are used to model the cause and effect relationship between the sending of the OUT signals and the PERT up to the microphone at PV.

## Graphics and Simulation in the Simulink Environment

To generate the model in *Simulink*, representing the noise is achieved using a sinusoid of unitary amplitude and frequency of 300 Hz for reasons of simplicity. The P & S systems are represented using second order models with similar response times. Both relations differ physically only in the system emitter and in the influence on the dynamic resulting from the difference in the distance of both the models, apart from the temporary displacement specified here through  $D_s$  and  $D_p$ .

In this case, it is assumed that Actual Input (AI) is directly the same as the OUT delayed by duration of the sampling period (ST). Figure 17 shows the proposed diagram:

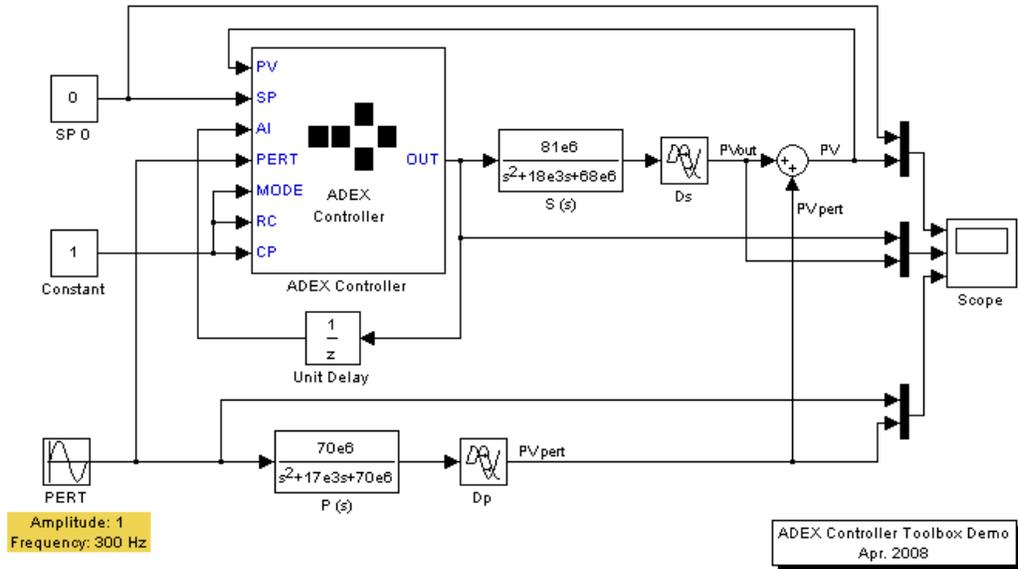


FIGURE 17. SCHEME

Taking into consideration the response times of the primary and secondary systems and the frequency of the perturbation, the following values have been selected:

- Sampling period for the controller (ST) equal to 80  $\mu$ s.
- Delay in the Primary channel, Dp of 160  $\mu$ s (more than 54mm).
- Delay in the secondary channel, Ds of 80  $\mu$ s (whose distance is equivalent to 27mm).

Once the controller is created, it needs to be able to read the delays. This can be carried out using the ADEX Writer Block or using the program ADEX COP Configurator, modifying the values DP, OUT and PERT respectively see (Figure 18). These DP's are written based on CP and ST.

In this particular case, the Control Period CP was configured to be the same as the sampling time ST, and as a result, the DP of OUT is 1, the DP of PERT is 3 (2 for the basic delay + 1 for conversion to digital which is considered internally by OUT).

With the Configurator, the delay times are changed (Figure 18):

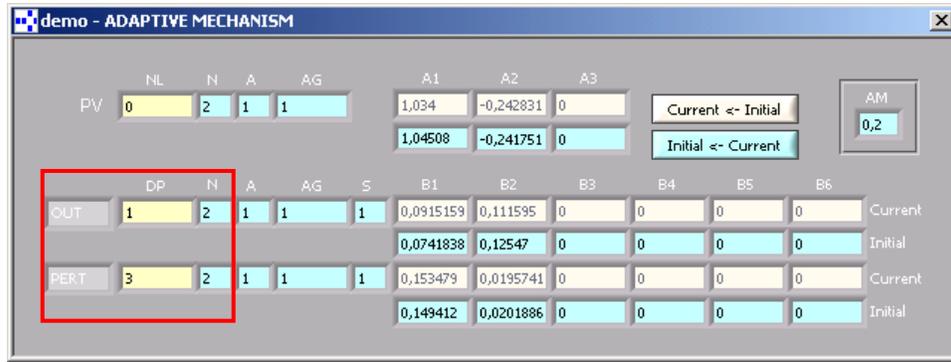


FIGURE 18. ADAPTIVE MECHANISM

The multiplexer blocks act as microphones or sensors to capture the signals and present them graphically in an oscilloscope screen.

Based on the considerations above regarding the time delays, the next step is to carry out successive simulations so that the ADEX controller builds up a model from process knowledge, at the same time as manipulating the values of the remaining parameters using the available methods by means of ADEX Reader and Writer or ADEX COP 2 Configurator.

After adjusting the parameters, we can achieve these results:

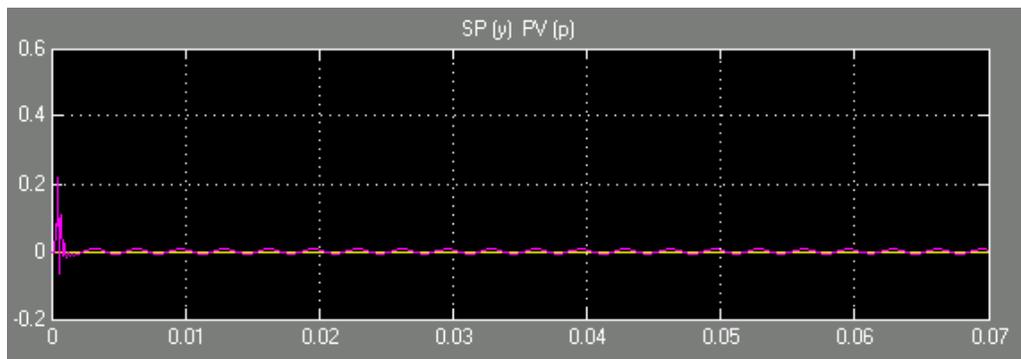


FIGURE 19. EXAMPLE RESULTS

As can be seen from the graphs in Figure 32, it is not possible to completely eliminate the noise, but the system was effective in reducing the effect by 150 times.

In Figure 20, there are represented in the upper graphic the control action and its effect in the output of this process; and the lower graphic represents

the disturbance and its impact on PV. Through this comparison can be seen how the signal OUT tries to counteract the effect of noise:

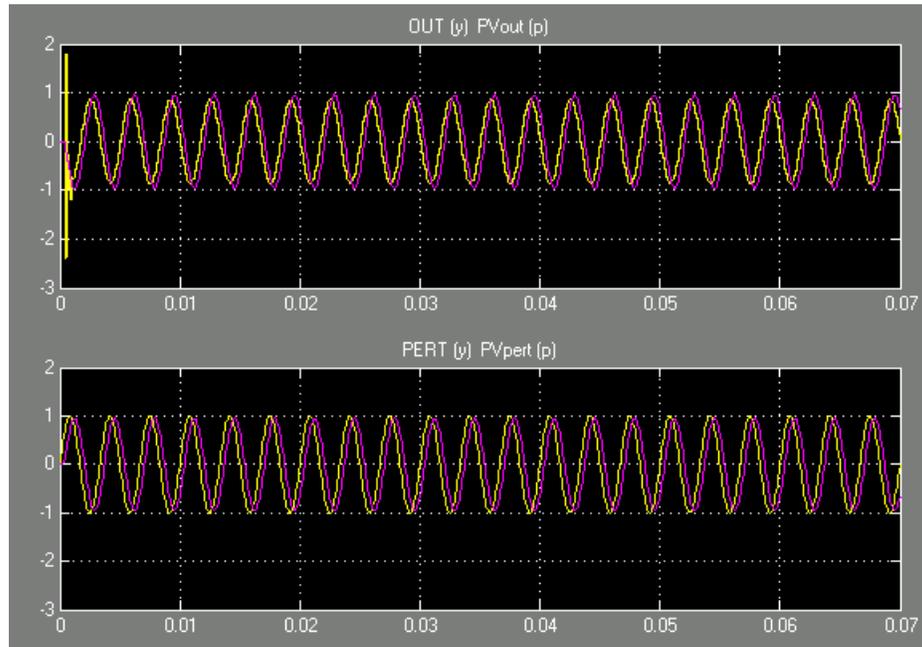


FIGURE 20. EXAMPLE RESULTS

This problem demonstrates the versatility of control schemes and additional options that can be achieved by means of using ADEX Controller Toolbox. This allows the user to integrate in *Simulink* automatic control strategies, designed for a specific process control.