MATLAB® Capabilities in Illustrating a Cost Effective Mini Hydro Power Plant Monitoring and Controlling Software

Amila PDG, Hettiarachi IT, Chandrasekera BLRN, Ameer MIS
Supervised by: Dr Lanka Udawatte
Department of Electrical Engineering, University of Moratuwa
sadam@sltnet.lk

***Abstract*- Main idea of the project is to introduce MATLAB as an automation software building tool while developing a monitoring and controlling system for a mini hydro power plant.**

I. Introduction

Under the fully automated environment visualization of the whole process of a mini hydro power plant is essential. Human Machine Interface (HMI) software need to be there to communicate with the process, control process and to present data gathered.

MATLAB has capabilities in providing the above requirements as well as satisfying our objectives. MATLAB is a high-performance language for technical computing. MATLAB is the tool of choice for high-productivity research, development, and analysis. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.[1]

Today companies like ABB, Alsthom dominate the Sri Lankan power sector with their costly software. Available automation building software includes comprehensive designing facilities though our finding (namely SIAR) tries to build a cost effective software using MATLAB with same capabilities.

SIAR facilitates the following functions.

1. Trend real-time and trend historical data.
2. Log data from process and write input to a database.
3. Monitor and controlling the process.
4. User interfaces.

II Methodology

SIAR gathers input signals through a Ni DAQ (National Instrument Data Acquisition Unit) and sound card. SIAR process the data through logics and pass to the interface for visualization.

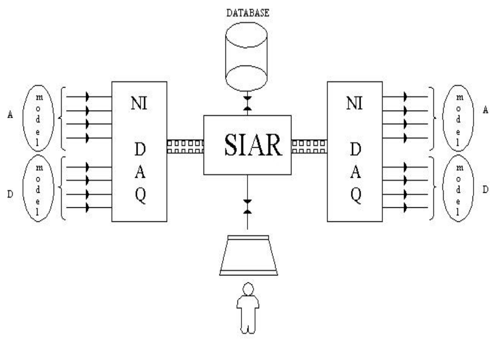

Fig 1: SIAR in controlling background

Following diagram shows the arrangement of the model.

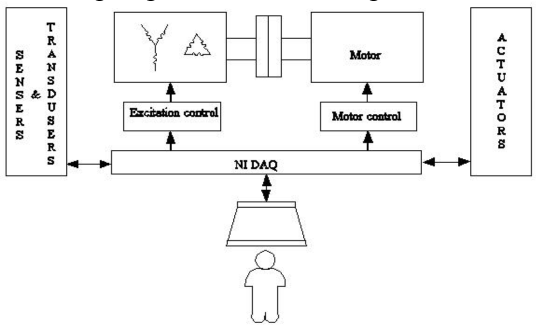

Fig 2: Model used for SIAR testing

The model consists of a synchronous generator coupled to a separately excited DC motor. The DC motor speed is controlled through the computer using a control circuit which controls the DC motor stator voltage.

III Hardware components

Analog inputs were collected from the soundcard. In this particular project we were in need of 3 analog signals to illustrate our software. They were current and voltage output of the generator and voltage of he grid. We can use the mic or line-in ports of the soundcard to collect the input. DSP toolbox has a block to collect the signal at given sample rates. Signals were transferred as frames of cycles. This made the computation of power, frequency etc. easier.

The data acquisition capability of the MATLAB environment was one of the main reasons we focused on when we choose MATLAB as our control system development software.

MATLAB's Data Acquisition Toolbox is a collection of M-file functions and MEX-file dynamic link libraries (DLLs) built on the MATLAB technical computing environment.[1]

In the SIAR system we decided to use both sound card and ni-daq acquisition support. We used the sound card for analog inputs and the ni-daq for digital inputs.

Acquisition through soundcard will involve two aspects sound card plugged and installed to the PC and the microphone and speaker will serve as the data acquisition (and output) devices.

Water level 'OK' signal is generated using the water level detector signal. A signal is passed from water level control unit whether minimum level is there or not.

In the actual system there exists the main inlet valve to control the speed of the alternator. In the system we are using instead of a MIV there exist a 220 V DC motor. Instead of controlling MIV the DC motor's speed is controlled. For that purpose we are using chopper control circuit.[2]

Initially we designed a control circuit using TL 084 to fire the thyristors and due to convenient we moved to an option of using a PIC.

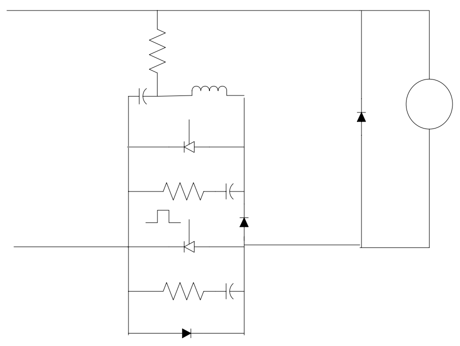

Figure 3 – Circuit Diagram of the Chopper circuit

When the load of the supply system changes the terminal voltage of the alternator also varies.

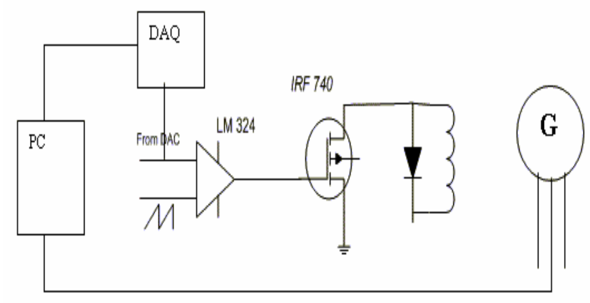

Figure 4 - Circuit diagram of the excitation Circuit

IV Software

A power plant is a complex data monitoring and manipulating environment. But the ultimate motive of an automated control system is to wrap around the complex data and provide the plant operating staff with user friendly data without them being directly involved with the plant floor complexities. The system basically consists of 8 windows (interfaces). They will have links between each other enabling the operator to view any required plant data without interrupting any other activity.
The figure shows the links between the interfaces.

1. Login Window
   The system will pop up with a start up window, which will request for predefined login information from the operator.
2. System Overview
   This window will give the major details of the plant at a glance to the operator and will consist of command button links to other important overview interfaces. The 'START' and 'STOP' button will enable the starting and stopping of the plant.
3. Starting Procedure
   Is used to start the generator and connect to the grid. This performs all pre requisites before starting the generator.
4. Single Line Diagram
   The Single Line Diagram will give an overview on the electrical installation of the plant and associated grid, the breaker status (open/closed) etc.
5. Stopping procedure
   Helps the user to follow the standard procedure of stopping the generator.
6. Reservoir Data
   This GUI will monitor the water level of the reservoir.
7. Generator Temperatures
   Temperature variations of the generator will be monitored to the operator and will pop up an alarm signal when ever a dangerous temperature level is detected over a pre defined safety level.
8. Data Plots
   With the above GUI's the operator will only get the real time instantaneous data of the plant floor. But this GUI will let the operator retrieve and plot visualization of past recorded data of the plant.

Logics

We were in need of critical software which will have the ability to implement our logics for the plant monitoring and control. This basically had to process the inputs, compare them with reference and output the required operation. We did analyze several applications to substitute our original intension of MATLAB 6.0. but ended up as it was in the original proposal.

Some of the other recent past methodologies was to fix everything in microcontrollers. But intension of this project was to analyze software logics instead. The disadvantages of microcontrollers are as follows.

- Once implemented the logics were fixed and it was complicated and difficult to make future changes. It is more effective to give an higher level of abstraction to the user where it would be maintainable.
- Devices and components are bulky and complex. Installation takes time and skills of highly technical personalities.

Although MATLAB was more used to implement system modeling, we used MATLAB here as a real time

A
r
m
a
t
u
r
e

system. The Digital Signal Processing toolbox was used for most of the processing as it was more on signal processing. Also the blocks used includes the Simulink, Data Acquisition etc.

Following is a demonstration of how we calculated the Active power for the purpose of monitoring the generator output.

Equation (1) corresponds to finding Active power with instantaneous current and voltage.

v(t) – instantaneous voltage (V)
i(t) – instantaneous current (A)
T - Cycle time (s)

$$\text{Power} = \frac{1}{T} \int v(t) \,.\, i(t)\, dt \qquad (1)$$

We can implement this equation using MATLAB blocks to obtain the power of the generator output. The implemented block is shown below.

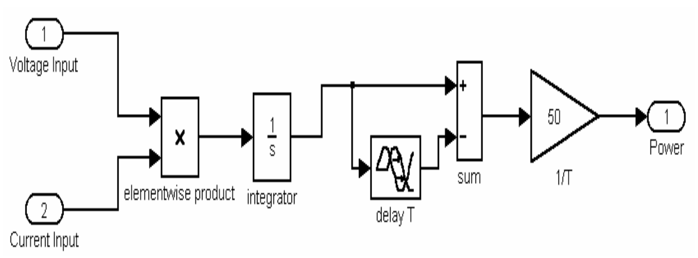

Figure 5– Implementation of Blocks to obtain power

First we take the element wise product of instantaneous voltage and current. Then we integrate over a single cycle and multiply by the fundamental frequency. This calculation of power is correct to any type of signal, irrelevant of whether it includes harmonics or not.

The above is an example of what we can implement using the already available blocks. But for computations which do not include such logics can be implemented by using further low level computations. These can be concatenated into a single block called the s-function. The block acts similar to any other block where the input is converted to output at specified time instants.

We implemented an s-function in calculating the frequency. Equation (2) shows how frequency is calculated as the number of samples between the maximum and minimum value averaged over 100 cycles.

t2,t1 - consecutive maximum detected times (s)

$$\text{Frequency} = 1 / (t2 - t1) \qquad (2)$$

MATLAB has extensive support in creating complicated blocks using basic operation blocks. Also you can abstract any operation as the number of blocks increase and get complicated. It is the operation of grouping the blocks into a subsystem. This would make the logic more understandable. For an example the following is our input signal filtering block which includes several other blocks such as low pass filter inside that block.

Basically the user is not allowed to view or edit any of the logics.

Starting Procedure

The power plant cannot be started instantaneously at any time as we wish. So the start sequence for the plant is specified in the SIAR system as below for the pre start checks.

Signal processing

The input signals received via soundcard has many distortions. We had to filter them to receive a pure sine wave of the voltage and current signals. The DSP block set had many such filtering components. These distortions include

- There was a input DC level of the soundcard power which had added to the signal. This was eliminated using the mean value at zero level.
- There was also several other noise which include processor fan and power supply fan noise and other noise generated inside the CPU. These were eliminated after calibrating the rms value of the input with measurements of the input signal.
- The above mentioned noise signals had very high frequencies of kHz range. We did attach a low pas filter to filter out as the signal we needed was within 50Hz.

Collecting input

Logics of Sound card data acquisition

There are a few steps that you must always follow; the steps initialize your hardware device and the channels you would like to sample. The ¯first thing you need to do is create an analog input device object and open up channels for data input.

```
AI = analoginput('winsound',0);
chan = addchannel(AI,1);
```

You have now defined the windows sound card to be the analog input device and have opened channel 1 for data input.

The analog input channel properties can be changed as suitable for out application.

In this example, we will take 44,100 samples in one second (this is a standard audio rate). We will trigger the hardware manually, which means that when we would like the sound card to start collecting data from the microphone when we type the command trigger(AI).

```
SampleRate = 8000;
set(AI,'SampleRate',SampleRate);
set(AI,'SamplesPerTrigger',duration*SampleRate);
set(AI,'TriggerType','Manual');
```

Now that we have configured the card you can begin acquiring data.

```
start(AI)
trigger(AI);
data = getdata(AI);
```

These commands work as follows: After the start command is issued the card is collecting data and recycling it through a temporary buffer. Viewing AI following the start should now display the Engine Status as 'waiting for TRIGGER 1 of 1'. After the trigger command is issued, the card is triggered and collects 8000 samples for one second. The data is pushed into the MATLAB workspace by issuing the get data command.

In the SIAR system without using the code we used the "from Wav device" block of DSP Blockset>>DSP Sources in MATLAB 6. This block is placed in a simulink model and the acquired data was manipulated in the model as wil be states in another section of the report.

Figure 6 – From wave device Block

We had to choose a sample rate of 8000Hz for our computations. A lower rate was unable to pick important points of the signal, which was required to calculate the frequency and filter out noises. A upper frequency was an overload of data which made the calculations slower and the extensive misuse of the input devices and memory.

Logics For Ni-Daq

1. Create a device object - Create the digital I/O object dio for a National Instruments PCI-6024E board with hardware ID 1.

```
dio = digitalio('nidaq',1)
```

2. Add lines - Add eight hardware lines to dio, and configure them for output.

```
addline(dio,0:7,'out')
```

3. Read and write values - Create an array of output values, and write the values to the digital I/O subsystem.
Note that reading and writing digital I/O line values typically does not require that you configure specific property values.

```
pval = [1 1 1 1 0 1 0 1]
putvalue(dio,pval)
gval = getvalue(dio)
```

4. Clean up - When you no longer need dio, you should remove it from memory and from the MATLAB workspace.

```
delete(dio)
clear dio
```

Calibration

The data acquired through the sound card is not proportional and they also contain some amount of noise. So the data should be filtered and calibrated to before analyzing or manipulating them. After the hardware and software are installed and the sensors are connected, the data acquisition hardware should be calibrated. Calibration consists of providing a known input to the system and recording the output. For many data acquisition devices, calibration can be easily accomplished with software provided by the vendor. But in the case of the sound card we had to give a known voltage signal step down below 1V and recorded the output in MATLAB workspace to find out the calibrating factors.

Databases

Storing data is one of the key issues that should be implemented in a control system. This will enable debugging or fault finding of a system and also will help the operator in forecasting useful information. Thus in the SIAR system a database is used to record the key data of the system. The database is Microsoft Access format. This contains several tables, namely Voltage, Active Power, Reactive Power, power factor, Current and Water level. The real time data is inserted into these tables as they are acquired.

The Database Toolbox is one of an extensive collection of toolboxes for use with MATLAB. The Database Toolbox enables you to move data (both importing and exporting) between MATLAB and popular relational databases. Data is retrieved from the database and stored in the MATLAB workspace. At that point, you use the extensive set of MATLAB tools to work with the data. You can include Database Toolbox functions in MATLAB M-files. To export the data from MATLAB to a database, you use MATLAB functions. The Visual Query Builder (VQB), which comes with the Database Toolbox, is an easy-to-use graphical user interface for retrieving data from your database.With the VQB, you can display the retrieved information in relational tables, reports and charts.

ACKNOWLEDGMENT

We shall be thankful to the Electrical Engineer of old Laxapana power plant Mr. Saumya Kumarawadu and Operational Engineer Mr. Shantha Kumara for there helping hand in building the idea of the project.

At the outset, our sincere gratitude goes to Engineer Mr Sudath Annasiwatta and Engineer Mr. Sudath Chandana and the staff of Lanka Transformers Ltd for all the assistance given for the project as the chief engineer of the Nivindu Power plant.

We shall be thankful to Dr Nishantha Nanayakara for sharing his knowledge in operation and control of mini hydro power plants.

I wish to extend my sincere thanks to Dr. Lanka Udawatta supervisor of the project for excellent guidance and directions. Sincere acknowledgement must go to machine laboratory staff of Department of Electrical Engineering for their enduring support.

REFERENCES

[1] MATLAB version 6.0.0.88 Release 12 help files – Simulink help, DSP help

[2] Nivindu power plant software operation manuals.