24.536 Reactor Experiments and 407.403 Advanced Nuclear Lab

Demo #0 Description/Procedure: Summary of the Data Acquisition Demo

Introduction

One of the first tasks that must be mastered in this course deals with accessing and working with the reactor data generated during the various experiments. This process is relatively straight-forward and it will be illustrated in detail during the first class. Thus, the summary overview/ instructions given here will be quite brief, primarily serving as a quick outline of the steps that are required each time you work with a new dataset.

In general you will participate in the reactor labs via the UMLRR Online application and via web-based conferencing with the reactor operators. After completion of the lab, the composite data from the full day's operation can be downloaded from within UMLRR Online and used with various Matlab-based offline visualization and data analysis tools to help extract and analyze the data pertinent for a given experiment -- and this is the procedure we will outline in this brief "How To" guide.

A Sample Experiment

We will use an actual experiment that was performed on August 16, 2012 as a specific example of the procedure. The goal of this specific experiment was to measure the combined reactivity feedback effect of both xenon and temperature during a 4-hour run while at near full power operation. The secondary cooling pump was turned off during the majority of the run -- so, with no cooling, the primary-side temperatures certainly increased during the run. In addition, the regulating blade was put in auto mode, and the RegBlade was observed to continually move out of the core over the duration of the experiment, while maintaining the reactor at the just critical state. The positive reactivity associated with the RegBlade moving out was needed to compensate for the negative reactivity associated with the buildup of xenon and the increase in the average core temperature (i.e. with the secondary pump off, all the primary-side temperatures increased nearly linearly during the experiment). The basic idea here was simply to measure and plot the feedback reactivity associated with the xenon buildup and increasing temperatures. The RegBlade worth curve is used to relate the blade position vs. time to the reactivity worth needed to compensate for the inherent feedback reactivity.

Summary Procedure

To perform the analysis outlined above, we need to perform the following tasks:

1. Download the **umlrr_data.hst** file using the ***Download*** screen within the UMLRR Online application. This file contains the history of all the data recorded during the current day's operation of the UMLRR (with a 1 sec sampling period). It is a binary file written in a proprietary format that cannot be accessed directly by the user in its present format. You should always rename this file appropriately to include the substance and date of the experiment. For the current example, the **umlrr_data.hst** file was renamed to **xenon+temp_test_081612.hst**. This demo will work with this archived history file.

   Note: The Download Screen function is currently Not Operational within the UMLRR Online application. Instead, the **umlrr_data.hst** file (with a new name) will be made available shortly after a given experiment via the course Dropbox share folder for the current semester.

2. A Matlab-based GUI called **umlrr_data** was written to process, visualize, and analyze the data from the UMLRR. The complete program is stored in the **data_gui_ver4.1p.zip** file in the Dropbox shared folder for this class. You should download this file to your computer and unzip all the files within a known directory location. The zip file contains a number of *.p files, a few *.fig files and a few additional support files. Open Matlab, ***put the folder with the data GUI within Matlab's path***, and then run the code by typing **umlrr_data** within the Matlab command window (the **umlrr_data.p** file is the main program which calls the other *.fig and *.p function files as needed).

   Note: A Matlab p-file is simply an encrypted form of a standard Matlab m-file. This was done to protect some proprietary data within the codes and to prevent the student from inadvertently making modifications to the code. All the Matlab programs with graphical user interfaces (GUIs) distributed as part of this course will be formatted in a similar fashion. These codes are intended to be used "as is" without any user modification.

3. Once you have started the GUI, two windows will appear -- one is the main visualization screen for viewing the reactor data and the other is a tags selection window, where you simply select or de-select the items to be plotted within a particular plot group. The GUI usage is pretty intuitive and its basic operation will be described and demonstrated in class, so only two details will be highlighted here, as follows:

   a. The first task to complete within the GUI is to open the desired history file with either a *.hst, *.txt, or *.dat extension. On first use, the user will open the binary *.hst file that was downloaded from UMLRR Online. Upon doing this, a DOS-based converter program (**hst2txt.exe**) is executed in the background that converts the original *.hst file into an ascii-formatted *.txt file that contains the bulk of the recorded reactor data, and a short *.hdr file that contains the titles (i.e. tag names) for each of the columns of data in the *.txt file. The file names for these two files are the same as the *.hst file (except, of course, for the file extension) and these should not be changed. Note that, in the current version of the data GUI, the **hst2txt.exe** program ***MUST be in the current Matlab folder*** -- so be sure you check for this if the data file does not appear to open properly. For our example, when the history file **xenon+temp_test_081612.hst** was read into the GUI, the **xenon+temp_test_081612.txt** and **xenon+temp_test_081612.hdr** files were generated. If these data are needed in subsequent use of the GUI, the *.txt file (and *.hdr file) should be used to avoid redundant and unnecessary use of the converter program.

   b. After browsing the available data with the GUI, one often wants to select data for only a subset of the recorded time interval for subsequent detailed analysis. Once the data range of interest is displayed in the visualization screen, the user can simply hit the ***Save Reduced Data File*** button to save the data within this interval (with whatever averaging interval was selected at the time) to another file with a *.dat extension. This new and usually significantly smaller *.dat file can be brought back into the GUI for later use or it can be read directly within a user-written Matlab program for specialized processing (this file can also be easily imported into Excel, if desired). As part of our example, we saved the reduced **xenon_temp_feedback_081612.dat** file in this fashion, and it contains data for only four hours starting at 10:03 am (or 11:03 am if "Daylight Saving Time" is in effect) with the sensor values averaged over 30 second intervals -- with a significant reduction in file size (from 11.7 MB for the *.txt file to 0.3 MB for the *.dat file). The

time interval chosen for study here was the constant power portion of the run (over 2 hours of testing and reactor startup operations was eliminated from the data file).

4. The final step is to read the *.dat file in a specialized user-generated Matlab program to do some further analysis as requested for each of the reactor labs. As an illustration of this step, the **fdbk_rho_081612.m** file was written to read the **xenon_temp_feedback_081612.dat** file, generate some basic plots of the blade positions and core temperatures vs. time and then, using some previously-generated information concerning the integral blade worth curve for the RegBlade, we generate a plot of the feedback reactivity vs. time (for this example, this was our ultimate goal!!!). The comments within the **fdbk_rho_081612.m** code make this program pretty easy to decipher and it is also easy to identify the various pieces of the code that can be re-used in student-generated code for the individual labs. Thus, this sample data processing demo should make it straightforward to perform these same types of tasks in future assignments.

Available Files

Many of the files needed to reproduce the in-class Data Acquisition Demo are available in the Dropbox shared folder for this class in the **xenon+temp_test_081612.zip** file and the **data_gui_ver4.1p.zip** file. With these data and program files, you should be able to practice doing some data processing on your own. Working out the details here early in the semester will make your subsequent assignments much easier. Good luck -- this should actually be fun and you might even learn a little about reactor operations as a side benefit...

Final Note: The sequence of steps outlined above are representative of the tasks that you will need to perform each time we complete a new experiment within the UMLRR. In summary, they include:

1. **Open** the ***.hst** file within the **umlrr_data** GUI, **study** the reactor data by viewing several of the available time-dependent plots (core power, various temperature sensors, control blade positions, etc.), and **carefully identify the time region of interest** (you can isolate the desired time interval with the **Start Time** and **Duration** edit windows within the GUI).
2. With the desired time interval properly selected, **save a reduced data file** for subsequent processing and analysis within a case-specific Matlab file.
3. **Write (or modify) a case-specific Matlab file** to do any desired analysis that may be required for the given experiment.

These general tasks will need to be performed several times over the course of the semester. Thus, please take the time now to be sure that you can perform these tasks with relative ease -- and please study the various code segments within the sample **fdbk_rho_081612.m** code so that you can use/modify these code elements, as needed, when you write your own Matlab codes to perform the desired post-processing tasks requested for each experiment. Note that student feedback from several previous classes has indicated that this is the most difficult part of this course (i.e. writing the Matlab post-processing codes), so take the time NOW to make sure that you get a good foundation here -- since doing this now will make life a lot easier as the semester progresses…