FIRST EXPERIENCE WITH THE MATLAB MIDDLE LAYER AT ANKA

S. Marsching*, aquenos GmbH, Baden-Baden, Germany
E. Huttel, M. Klein, A.-S. Müller, N.J. Smale, KIT, Karlsruhe, Germany

Abstract

The MATLAB Middle Layer has been adapted for use at ANKA. It was finally commissioned in March 2011. It is used for accelerator physics studies and regular tasks like beam-based alignment and response matrix analysis using LOCO. Furthermore, we intend to study the MATLAB Middle Layer as default orbit correction tool for user operation. We report on the experience made during the commissioning process and present the latest results obtained while using the MATLAB Middle Layer for machine studies.

INTRODUCTION

Synchrotron light sources are an excellent tool for a wide range of studies in biology, chemistry, physics and medicine, as they provide very bright light from hard x-rays to the far infrared.

Most of these light sources are based on electron storage rings, in which electrons circle around at relativistic speed, thus emitting synchrotron light.

For ensuring a maximum performance at the generation of synchrotron light, it is crucial to have tight control over the various operational parameters of the underlying particle accelerator and adapt to changing demands or environmental conditions, like the air temperature.

The MATLAB Middle Layer (MML) [1] is a collection of scripts for the MATLAB programming environment, designed to control and measure parameters of an accelerator. This software creates an abstraction layer enabling the user to write scripts independent of the accelerator's control system. For most uses, even the magnet lattice of the accelerator itself is abstracted. Thus, scripts written for one accelerator can be used for a different accelerator, even if this accelerator has a mostly different design.

Recently, we have adapted the MATLAB Middle Layer for use at ANKA. ANKA is the synchrotron radiation facility of the Karlsruhe Institute of Technology, located in Southern Germany.

In order to use the MML at ANKA, we had to make an interface to the ANKA control system. First we implemented our own control system interface in the MML. Later we migrated to the built-in LabCA support and used a gateway to make data from our control system available via the Channel Access (CA) protocol.

We use some of the existing tools that are part of the MML, like code for orbit response matrix and dispersion measurements as well as the orbit correction code. However, we extended the code for beam-based alignment to provider better fit results and thus improved data for the beam position monitor (BPM) offsets. Finally, we wrote some code to wrap the mostly used functions of the MML in a nice graphical user interface (GUI).

* sebastian <dot> marsching <at> aquenos.com

CHOOSING THE MATLAB MIDDLE LAYER

ANKA has had a software that links the control system to a physical model of the accelerator for quite a long time [2]. However, this software was completely implemented in Java, and thus did not provide the simple scripting mechanisms of the MATLAB Middle Layer. Therefore it was only used for some specific applications (e.g. the orbit correction code still in daily use), but was not used by the accelerator physicists to write their own code.

Learning from this experience, a software for linking the ANKA control system to the MATLAB-based Accelerator Toolbox was developed [3]. Using this software, physicists could finally write simple MATLAB scripts for controlling the accelerator. However, the software suffered from the lack of existing application code. As it was specifically written for ANKA, code from other accelerators could not be used without changes. Thus this software's benefits for daily operation were limited.

Therefore, we decided to adapt the existing MATLAB Middle Layer code for ANKA, reusing parts of the control system link written to connect MATLAB to the ANKA control system. This way, we could link the advantages of having an interface for simple MATLAB scripting with the advantages of already having a number of useful applications.

ADAPTING FOR ANKA

In order to adapt the MATLAB Middle Layer for use at ANKA, we had to create the data-structures specific to the accelerator's lattice and we had to implement a link to our control system.

Creating the Data-Structures

For adapting the MML for a new accelerator some data-structures have to be generated. While their structure is the same for all accelerators and can just be copied, their content (e.g. magnet calibration curves and names of control system properties) has to be customized. The total effort for finishing this task did not exceed a few days.

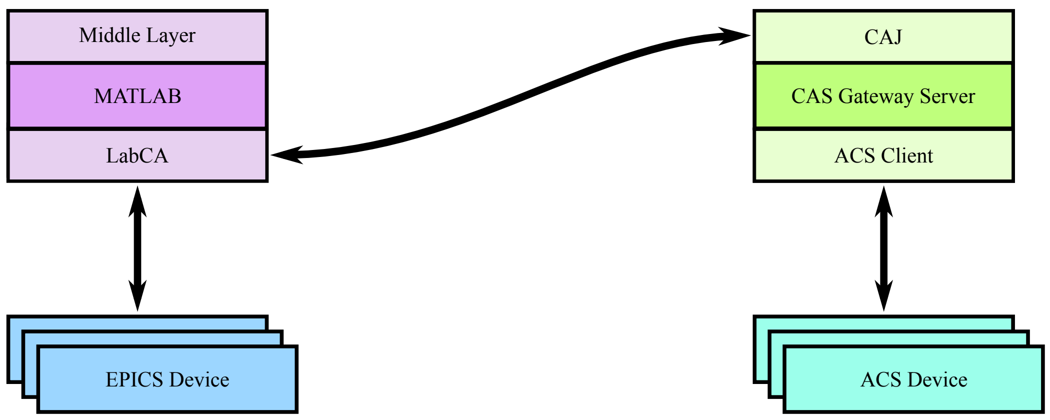


Figure 1: Structure of the control system access for the MATLAB Middle Layer at ANKA. MATLAB communicates directly with EPICS devices using the LabCA library. ACS devices are accessed through the CAS Gateway Server specifically developed for ANKA.

Link to the Control System

At ANKA, most of the devices relevant for the MML (mainly power supplies and beam position monitors) are controlled via the ALMA Common Software (ACS) [4] control system. The MML had no built-in support for this control system, thus we had to develop our own control system link.

As integrating an ACS client directly into MATLAB was not feasible [3], we wrote a server based on the Hyper-Text Transfer Protocol (HTTP) [5]. Each property of the control system would be mapped to a unique resource identifier (URI) identifying this property. A property could then be read by sending a HTTP GET request to the server and written to by sending a HTTP POST request. The actual data would be encoded in the request's or response's body using the JSON protocol [6], a simple, text-based platform-independent protocol for structured data. This solution proved to work well and minimized the implementation effort in MATLAB.

Later, a new beam position monitor (BPM) system based on the Libera Brilliance device manufactured by Instrumentation Technologies was acquired by ANKA. However, this system was not integrated into the ACS-based control system, but uses existing EPICS software developed at DIAMOND [7].

Adding EPICS support was not trivial, because the control system link support of the MML basically depends on two specially named functions (getpvonline and setpvonline). As there is no way for specifying different versions of these functions for different control system properties, we could not use EPICS- and ACS-based devices from the MML in parallel.

We identified two feasible alternatives for integrating both systems in parallel: We could either extend our HTTP server to connect to EPICS-based devices or we could use one of the existing Channel Access (CA) implementations included in the MML and talk to the new BPM devices directly. In the latter case, we would need to use the CA protocol for communication with the legacy control system as well.

We chose the second option for mainly three reasons: First, we would not have another layer of software for the communication with the new BPMs, thus increasing the reliability and performance. Second, using the CA protocol instead of HTTP promised a slightly better performance, because this was a binary protocol specifically designed for control system communication. Finally, we anticipated that in the future more hardware at ANKA might use EPICS instead of ACS and therefore the first argument might become more and more important.

Therefore, we changed our gateway software to provide a Channel Access server instead of an HTTP server, resulting in the setup depicted in Fig. 1. We could reuse major parts of the code communicating with ACS and used the Channel Access for Java (CAJ) [9] library for implementing the server side.

IMPROVEMENTS FOR BEAM-BASED ALIGNMENT

The MATLAB Middle Layer brings a set of functions for running a beam-based alignment (BBA). This is a method to determine the offset of BPMs by changing the magnetic field of a close-by quadrupole for different orbits. These different orbits are set by changing a single corrector magnet. When the orbit does not change when the quadrupole field is changed, the beam is in the center of the quadrupole and thus regarded to be in the center of the close-by BPM.

The quadplot function included in the MML determines the BPM offset by fitting a linear or quadratic function to the orbit change at each BPM for the different position at the aligned BPM. Figure 2 shows the fit for real data measured for BPM S2.01 (horizontal) at ANKA.

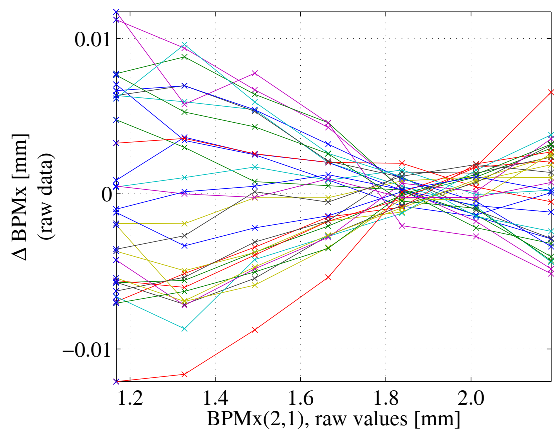


Figure 2: BBA data evaluated with the quadplot function. Apparently the BPM center is at about 1.9 mm. However, due to the noisy data, quadplot miscalculates the BPM center to be at 0.83 mm.

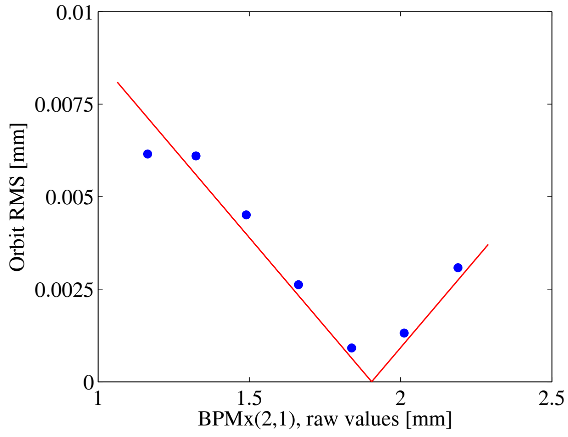


Figure 3: BBA data evaluated with the quadplot2 function. The measured data is shown in blue, the fit is shown in red. Using the same data as in Fig. 2, this function calculates the BPM center to be at 1.90 mm, which is in the expected range.

However, this fitting method is sensitive to single defective values and does not work nicely if the measuring range is far off-center. Therefore we wrote a function called quadplot2, implementing a different algorithm:

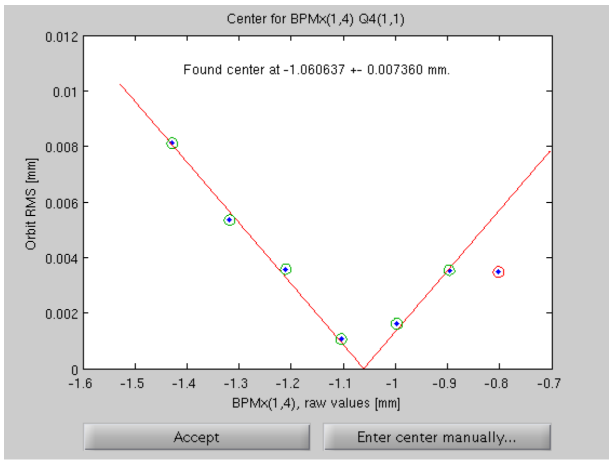
This function does not regard single BPMs but calculates the root-mean-square (RMS) of each orbit difference. This RMS value is plotted over the beam position measured at the BPM to be aligned. Subsequently we fit a function of the form

$$f(x) = |x - a| \cdot b, \tag{1}$$

where x is the beam position at the BPM to be aligned, $f(x)$ is the RMS of the orbit difference for the respective orbit, a is the BPM offset and b is a free fit parameter.

The result of the fit for the same data as in Fig. 2 is depicted in Fig. 3. For the example data, this algorithms improves the result, because it is less sensitive to the asymmetry of the measured data and less sensitive to noise.

Still, single data-points used for the fit can be invalid, e.g. because the change for the corrector magnet sent to the control system has not been applied correctly. Therefore, we added an interactive mode, where the user can see the measured data and the fit and select or deselect individual data-points to be included in the fit. In this way, the user can remove obviously faulty data-points and gets an idea of how good the fit results are. The improvements achieved by using this interactive mode for a partly defective measurement are shown in Fig. 4.


Center for BPMx(1,4) Q4(1,1)

Found center at -1.060637 +- 0.007360 mm.

Figure 4: Interactive fitting GUI. The last data-point has been deselected, because it is obviously flawed.

USER-FRIENDLY GUI

In order to make the frequently used functions of the MML available to those users who do not want to learn how to use the command-line interface, we created a simple GUI which serves as a starter for these functions. Figure 5 shows the start panel, which then opens sub-panels for the different functions like dispersion and orbit-response-matrix measurements, beam-based alignment and orbit correction.

This panel is even useful to those users who know the command-line interface, but just want to quickly run a measurement without having to look up the exact syntax of the corresponding function.

THE MATLAB MIDDLE LAYER IN ACTION

As important measurement tools like orbit-response-matrix and dispersion measurements are provided in a standardized way by the MML, it very easy to use LOCO [10], a tool for analyzing the linear optics of an accelerator using orbit-response-matrix measurements. In fact LOCO is already included in the MML distribution and only a few

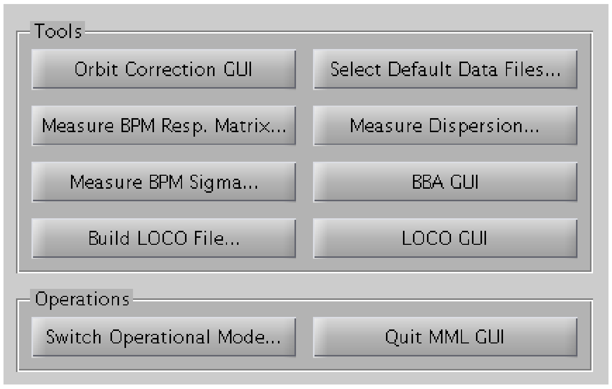


Figure 5: Starter GUI panel for the MML at ANKA.

accelerator-dependent parameters have to be adjusted.

One of our first analyses using LOCO and the MML revealed, that the kick strengths used for measuring the response matrix strongly dependend on the corrector magnet.. The kick strengths fitted by LOCO are shown in Fig. 6.

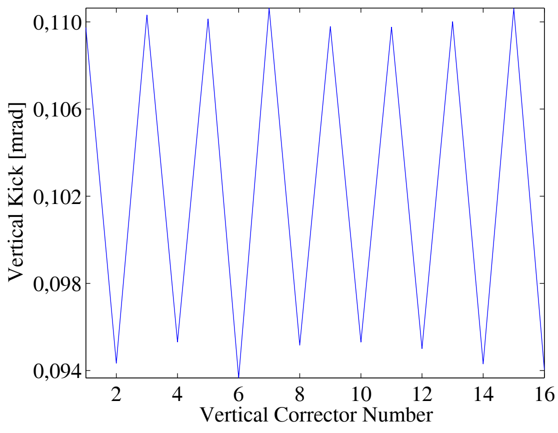


Figure 6: Vertical corrector kicks fitted using LOCO.

First, because of the alternating structure, we suspected numerical noise causing a defective fit. However, closer investigations revealed, that this was caused by an error in the calibration curves converting from magnet current to kick strength and vice versa.

For geometrical reasons, there are two types of vertical corrector magnets at ANKA. Both are expected to have a slightly different kick strength of 1.485 mrad A^{-1} GeV^{-1} and 1.650 mrad A^{-1} GeV^{-1} respectively.

However, as we now discovered using LOCO, in reality both types seem to have the same kick strength of about 1.5 mrad A^{-1} GeV^{-1}.

CONCLUSION

We commissioned the MATLAB Middle Layer for use at ANKA. The commissioning process was quite simple and would have been even simpler if we had used one of the control systems directly supported by the MML.

We now frequently use the MML for machine physics studies and are investigating the option to use an MML-based orbit correction code for regular operation, too. Physicists at ANKA writing their own code for machine physics measurements have already started using the MML to get information about the state of the accelerator.

The alternative BBA code we have written shows significant advantages over the code distributed with the MML in handling imperfect measurement data. The MML simplifies the task of performing a beam-based alignment dramatically compared to our old solution which required the user to copy the measured data into Microsoft Excel for further evaluation.

By creating a GUI for the most commonly used functions, we could even convince those users to use the MML, who are not familiar with the Middle Layer's MATLAB command-line interface.

In summary, adapting the MATLAB Middle Layer for use at ANKA has already brought significant benefits within the first few months of use and we are looking forward to exploit the features of the MML to an even larger extent.

REFERENCES

[1] G. Portmann *et al.*, "An accelerator Control Middle Layer using MATLAB", PAC'05, Knoxville, May 2005, p. 4009, http://www.jacow.org.

[2] I. Kriznar *et al.*, "Cross-Platform and Cross-Accelerator Machine Physics Programs with Databush", ICALEPCS'03, Gyeongju, October 2003, p. 151, http://www.jacow.org.

[3] S. Marsching *et al.*, "A High-Level Interface for the ANKA Control System", ICAP'09, San Francisco, August 2009, p. 318, http://www.jacow.org.

[4] ALMA Common Software website, http://www.eso.org/projects/alma/develop/acs/.

[5] R. Fielding *et al.*, "Hypertext Transfer Protocol – HTTP/1.1", IETF RFC2616, June 1999, http://www.rfc-editor.org.

[6] D. Crockford, "The application/json Media Type for JavaScript Object Notation (JSON)", IETF RFC4627, July 2006, http://www.rfc-editor.org.

[7] M.G. Abbott *et al.*, "The DIAMOND Light Source Control System Interface to the Libera Electron Beam Position Monitors", ICALEPCS'09, October 2009, Kobe, p. 694, http://www.jacow.org.

[8] T. Straumann, "labCA – An EPICS Channel Access Interface for scilab and matlab", May 2008, http://www.slac.stanford.edu/comp/unix/package/epics/extensions/labca/manual/.

[9] Channel Access for Java website, http://epics-jca.sourceforge.net/caj/.

[10] J. Safranek *et al.*, "MATLAB-based LOCO", EPAC'02, Paris, June 2002, p. 1184, http://www.jacow.org.