HIGH LEVEL MATLAB APPLICATION PROGRAMS FOR SPEAR3[§]

J. Corbett *et al*, SLAC, Stanford, CA 94309

Abstract

The SPEAR3 control system nominally operates with the EPICS toolbox on top of VMS hardware. The simultaneous use of Matlab Middlelayer (MML) and Accelerator Toolbox (AT) allow for parallel, high-level machine control and accelerator physics applications that communicate with the control system via EPICS Channel Access (LabCA). While the majority of the MML and AT software is machine independent, site-specific high-level applications are also required to control the accelerator. This paper describes several such high-level application programs that have been developed for control and diagnostics at SPEAR3. Examples include a time-dependent waveform display gui, beam steering applications, transport line optics correction, SR beam diagnostics and add-ons to the main MML routines.

INTRODUCTION

The SPEAR3 light source came as the result of a Basic Energy Sciences committee recommendation following a review of U.S. synchrotron radiation facilities in 1997 [1]. Before formal DOE/NIH funding arrived in 1999, preliminary lattice design and system engineering specifications were developed on project seed money. During this time, it became clear that the historical, yet dated, SPEAR control system would need to be largely replaced [2], in particular the high-level application programs. The new system would utilize EPICS operating on a VMS platform which opened up the possibility for Channel Access communication with external programs. In order to consider options for modern application development platforms, a satellite meeting was arranged at the 1998 International Computational Accelerator Physics Conference in Monterey, CA. Presentations included options for SDDS, TCL/TK and X-Windows software.

At the time of the Monterey conference, Matlab was already in use at SSRL for data processing and off-line accelerator physics calculations. Matlab had also been used extensively at the SLC for data acquisition, data reduction and to some degree machine control. At the ALS in Berkeley, Matlab was in use for command-line driven machine control and data processing [3], and had the interesting feature that the top-level language closely mimicked accelerator simulation programs such as TRACY [4]. At the same time the first versions of the Matlab Accelerator Toolbox [5] utilizing TRACY transport physics were available for simulation studies at SSRL.

During the Monterey meeting, a proponent of IDL made an interesting observation – since recent versions of Matlab contained graphical interface commands why not use it to develop high-level application programs [6]? With Channel Access connectivity embedded in Matlab (LabCA) [7], a complete solution was available with control system communication, gui capability, user-friendly data reduction software and accelerator simulation tools that could be integrated into a single, all-in-one software package. The gavel fell and a new project was born – high level application programs at SPEAR3 would be developed and written in Matlab[†].

In a stroke of luck, the main author of Matlab Middle Layer (MML) [8] was finishing work on an SBIR grant at SLAC and was available to consult with SSRL on application development for SPEAR3. The first project was to convert the FORTRAN version of the Linear-Optics-Closed-Orbit (LOCO) program to Matlab [9]. It was then recognized that SPEAR3 needed a 'middle layer' to provide easy connectivity between the accelerator physicist and storage ring. By introducing Matlab code utilizing accelerator modeling syntax developed at the ALS, a straight-forward database-drive system was devised for simulation and control.

As more of the ALS software was integrated into the system, the functionality of higher-level programs such as, orbit, tune, dispersion and chromaticity measurement expanded. In order to retain the ability to pass the new software back to the ALS, programs were written in a 'machine independent' format driven by simple MML initialization files to associate accelerator elements and their indices with girder locations, database channel names, hardware limits, conversion factors and specific locations within the AT lattice file.

First tests of machine independence were made in trials at the Canadian Light Source and then again at the ALS. Interestingly, machine-independence also created a structural rigor within the software that ultimately simplified high-level program development and streamlined switching between on-line and simulation control modes. Hardware-to-physics conversion factors also enabled the user to 'switch' between hardware (e.g. amps) and physics (e.g. m^{-2}) units with a single command. Similarly, the AT lattice pointers automate switching between on-line and simulation modes with a single command. File directory specifications were then incorporated to automate data file look-up and data storage needs for machine control and simulation.

In the sections to follow we describe high-level application program developments at SPEAR3 in the areas of waveform variable display, main ring and transport line machine tuning and optical diagnostics.

[†]the philosophy was, and still is, 'anything that can be written in EPICS will be written in EPICS'.

[§]Work supported by US DOE Contract DE-AC03-76SF00515 and Office of Basic Energy Sciences, Division of Chemical Sciences.

CONTROL/DISPLAY APPLICATIONS

One of the first high-level MML application programs was a real-time orbit monitor program intended to compliment a manual orbit control program. It was soon recognized that the x- and y 'orbit' families were only two instances of a larger class of 'families' defined in the MML initialization files. By simply redefining the 'family' and the associated axis coordinates within the display, a broad selection of accelerator hardware families could be displayed in the same graphical interface. 'PlotOrbit' was converted into 'PlotFamily' including a complete set of options to display data in terms of absolute or relative values, and with interactive axis scaling features. By utilizing the built-in functionality available in the MML, 'saved' and 'golden' family data could be easily recalled into the graphical display.

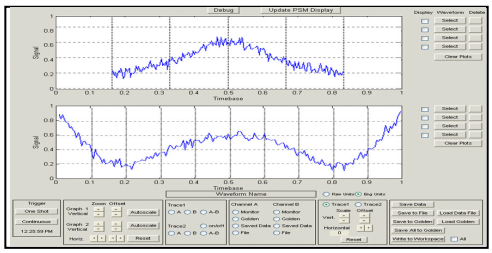

Figure 1: PlotWaveform graphical interface.

To further expand the PlotFamily interface, callbacks from main pull-down menus at the top of the display were programmed to execute other high-level MML code. This feature gave machine operators the capability to load and save entire machine configurations, measure and save machine parameters (dispersion, tune, chromaticity, LOCO data), and a means to control global accelerator properties (orbit, tune, etc). Graphical data could also be exported to the main Matlab workspace for further processing. The resulting PlotFamily application was machine-independent and could operate at any accelerator configured to run MML.

As an extension of the built-in functionality, PlotFamily has an added file execution option that executes at run time. This feature is used at SSRL to generate SPEAR3-specific menu options for transport line control, orbit-interlock checks, machine-specific diagnostic controls and links to hardware documentation.

A further development undertaken at SSRL was to incorporate the PlotFamily display features into a new, high-level 'PlotWaveform' graphical interface. As shown in Fig. 1, PlotWaveform provides a means to display real-time EPICS 'waveform' variables. Most of the ~50 EPICS waveform variables at SPEAR3 are supplied by the Pulse Signal Monitor (PSM) system which consists of a distributed set of analog signal amplifiers and digitizer boards to monitor pulsed RF data (few μs), fast-kicker data (few μs) and booster ramp signals (few ms). Similar to the MML initialization concept, PlotWaveform is based on a machine-specific initialization file that identifies common waveform names with Channel Access names, physical units and time base parameters.

MACHINE CONTROL APPLICATIONS

An early Matlab application program developed for SPEAR3 was the SVD-based orbit control interface 'OrbitGUI' [10]. The control interface utilized Matlab graphic features such as select-and-drag for beam position monitor icons while the underlying software utilized the MML library to open the corrector-to-bpm response matrix file, measure the beam orbit and load both RF frequency and corrector setpoints. The OrbitGUI program was nearly machine-independent with local specifics related to the fact that the code pre-dates MML.

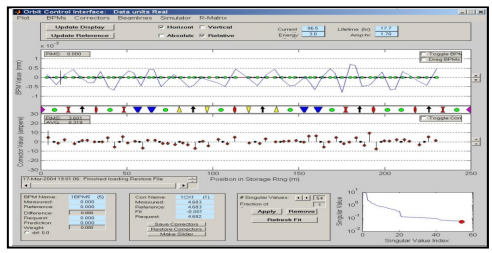

Figure 2. OrbitGUI graphical interface.

The main processing algorithm within OrbitGUI was then transferred to a slow orbit-correction feedback application (SOFB), which uses a Matlab timer object as the internal clock. The SOFB interface is more compact than the OrbitGUI interface allowing only timer on/off and RF correction on/off control. The internal SOFB orbit correction algorithm was updated to allow eigenvector-by-eigenvector mode discrimination. In this case, at each correction cycle SOFB calculates the inner product between the orbit vector and each orbit basis vector in the linear algebra sense. The correction is only applied if the inner product exceeds a pre-specified threshold for each mode. Operationally the discrimination algorithm better rejects BPM noise and results in a quieter beam orbit at the user beam lines.

A similar interactive orbit control program was developed for the linac-to-booster transfer line (LTB). In this case the response matrix is for 'open' as opposed to 'closed' beam orbits and the BPM data requires averaging for accurate results. In order to reliably steer the beam through the LTB, the initial launch conditions (x,x',y,y',dp) must be measured and held constant to minimize mis-steering and dispersion generated upstream. LTBOrbitGUI and the associated response matrix measurement software utilize MML commands are fully integrated into the MML file directory system.

For the booster-to-storage ring (BTS) transfer line, a more complex software system was developed to calibrate the beam line quadrupole optics using LOCO-style response matrix calculations [11]. The BTS software also contains a steering package designed to optimize beam injection efficiency into SPEAR3 .

The RF bucket select software was originally implemented in Matlab but then converted into an EPICS control panel. The conversion was consistent with the philosophy that straight-forward machine-critical software should be written in EPICS where possible.

DIAGNOSTIC APPLICATIONS

The Matlab middle layer and Channel Access connectivity are also utilized for SPEAR3 optical diagnostics. The x-ray pinhole camera, for instance, acquires the beam image with a PointGrey CCD Flea camera [12] routed through IEEE-1394b Firewire to a standard PC. A software link to the PointGrey camera control library maps the image into Matlab memory for processing and display [13]. The Accelerator Toolbox is used to compute relevant betafunctions at the x-ray beam source point. In the nominal beam monitoring mode the measured beam parameters are written to EPICS using LabCA. During periods of machine development, MML scripts are used to manipulate electron beam position, coupling and emittance as measured by the pinhole camera. A Matlab script developed at the CLS calculates spectrally-integrated Fresnel diffraction integrals to characterize beam propagation from source to screen [14].

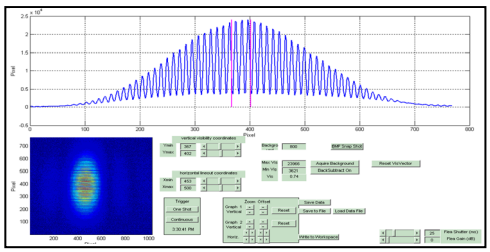

Figure 3: PlotWaveform graphical interface.

A similar, more sophisticated application program was developed for the visible-light interferometer [15]. As shown in Figure 3, a second Matlab-linked Flea camera acquires the raw, 2-slit interference pattern and a graphical interface is used to establish user-defined boundaries for the line-out. A Levenberg–Marquardt numerical fitting algorithm written in Matlab [16] applies a least-squares fit of a sinc/sine function to the interference data to extract the incoherent beam visibility. MML is again used to control insertion device parameters, x-y coupling and emittance for machine characterization.

For the fast-gated and streak cameras, direct links are not available to the internal camera software so raw camera images are saved to disk and re-opened in Matlab for processing. Moments of the transverse and longitudinal beam distribution are fitted to extract data relevant to emittance, machine impedance and instability thresholds. In cases where time-dependent phenomena are recorded, images are pre-processed and then sequenced together in Matlab to generate 'movies' that display non-linear features of the beam distribution that are otherwise difficult to characterize with scalar quantities.

SUMMARY

The Matlab middle layer has provided a relatively user-friendly software package for machine commissioning, operation and accelerator development. Key components include the Accelerator Toolbox, Channel Access Toolbox and a wide range of accessory tools. High-level application programs built largely on the MML allow for scripted data acquisition, data processing and graphical display that are difficult to implement using standard accelerator control system software. To date, over a dozen synchrotron light sources have adopted MML and many have gone on to develop high-level application programs. High-level application programs at SPEAR3 include waveform analysis, beam tuning and orbit control and optical diagnostics. Another important feature of MML and high-level application programs is the ability to provide teaching tools for students and interns.

REFERENCES

[1] R. Birgeneau, et al, 'BESAC Advisory Committee Panel on D.O.E. Synchrotron Radiation Sources and Science', November 1997.
[2] H. Rarback. et al, 'Old Wine in New Bottles-The SPEAR Control System Upgrade', ICALEPCS'99, October 4-8, Trieste, Italy, 1999.
[3] G. Portmann, 'ALS Storage Ring Setup and Control Using MATLAB', LBL LSAP Note #248, June 1998.
[4] H. Nishimura, 'TRACY, A Tool for Accelerator Design and Analysis', EPAC'88, Rome, Italy, 1988.
[5] A. Terebilo, 'Accelerator Modeling with MATLAB Accelerator Toolbox', PAC'01, May 2002.
[6] Harvey Rarback, private communication.
[7] A. Terebilo, 'Channel Access Toolbox for MATLAB', ICALEPCS'01, San Jose, CA, 2001.
[8] G. Portmann and J. Corbett, 'An Accelerator Control MiddleLayer Using Matlab, PCaPAC'05, Hayama, Japan.
[9] J. Safranek, et al, 'Linear Optic Correction Algorithm in MATLAB', PAC'03, Portland, Oregon, 2003.
[10] J. Corbett, 'Orbit Control Using MATLAB', PAC'01, Chicago, Illinois, 2002,
[11] J. Safranek, et al, 'Optimization of the Booster to SPEAR Transport Line for Top-Off Injection', PAC'09, Vancouver, Canada, 2009.
[12] http://www.ptgrey.com/products/flea2/index.asp
[13] Henrik Loos, LCLS, private communication.
[14] Jack Bergstrom, CLS, private communication.
[15] J. Corbett, et al, 'Interferometer Beam Size Measurements in SPEAR3', PAC'09, Vancouver, Canada, 2009.
[16] Xiaobiao Huang, SSRL, private communication.